ARPA Order No. 2223 ISI/TM-79-16
January 1979

6 B.S.

the JOY of TENEX and TOPS-20

... in two parts

PART TWO LEVEL II

* XED REFERENCE MANUAL ... a text editing system that has special-purpose commands for message creation and transmission, and text formatting/justification. XED also allows the user to customize XED's actions and responses by means of a user-settable mode feature.

* INTRODUCTION TO TECO ... a text editing system for basic, large-scale, and special text editing. The text which TECO edits is a pure character string; several commands can be typed in as a single command string.

* DCOPY - DIABLO PRINTER TERMINAL PROGRAM ... is a program created at ISI to handle the production of output on the DIABLO Printer Terminal. DCOPY is available on the ISI-TENEX and ISI-TOPS-20 operating systems.

* INTRODUCTION TO XOFF ... a document formatting program that generates files for output to a terminal, a line printer, or a Xerox Graphics Printer.

* The RUNFIL PROGRAM ... provides a means for using a file instead of a terminal to supply an input command stream.

* TENEX CALENDAR PROGRAM ... for entering, updating, modifying and encrypting typical calendar data (appointments, deadlines, and tasks) and providing reminders specified by the user.

* FILE TRANSFER PROTOCOL (FTP) ... provides facilities for file transfer between hosts on the ARPANET.

* TELNET USER GUIDE ... for communicating with host computers via the ARPANET utilizing the TELNET protocol.

* RESOURCE SHARING EXEC (RSEXEC) ... provides an environment in which the range of many features found on a single-host time-sharing system are extended beyond the boundaries of a single host to encompass many hosts on the ARPANET.

Prepared by
Chloe Sommers Holg

UNIVERSITY OF SOUTHERN CALIFORNIA
INFORMATION SCIENCES INSTITUTE
4676 Admiralty Way Marina del Rey California 90291

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ISI/TM-79-16 (14) (moved)
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: The Joy of TENEX and TOPS-20: Part Two II (6)
5. TYPE OF REPORT & PERIOD COVERED: Technical Manual (9)
6. PERFORMING ORG. REPORT NUMBER
AD69518
7. AUTHOR(s): Chloe Hola (10) Chloe Summers/Hola
8. CONTRACT OR GRANT NUMBER(s): DAHC 15 72C0308 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS: USC/Information Sciences Institute ✓, 4676 Admiralty Way, Marina del Rey, CA 90291
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (11) Jan 79
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209
12. REPORT DATE: March 1979
13. NUMBER OF PAGES: 112 (12) 113 p.
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: -----
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

This document approved for public release and sale; distribution unlimited.

(15) DAHC15-72-C-0308, VIARPA Order-2223

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

AD69518

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ARPANET, DCOPY, FTP, RSEXEC, RUNFIL, TECO, TELNET, TENEX, TOPS-20, XED, XOFF

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This is a basic manual discussing the operation of the ARPANET, the TENEX and TOPS-20 operating systems, and the following programs: XED, TECO, DCOPY, XOFF, RUNFIL, FTP, TELNET, and RSEXEC.

79 06 05 040

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

407 952

ARPA Order No. 2223
ISI/TM-79-16
January 1979

the JOY of TENEX and TOPS-20 in two parts

PART TWO

This document is one of a series of user manuals
being made available by ARPA IPTO through ISI.

Prepared by Chloe Sommers Holg
USC/Information Sciences Institute

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC 15 72 C 0308, ARPA Order No. 2223.

CONTENTS

1. XED REFERENCE MANUAL

1.1 HELP

H(elp)

H(elp) accepts a command character following the H command. A description of the command represented by that character is then printed on the terminal. H(elp) followed by a carriage return additionally requests a file name. A brief reference manual is created in the file.

The Legal Command Letters are:

```
↑A,↑H,DEL Delete char   ↑W Delete word  ↑X Delete line  ↑R Re-type line
↑Q Abort command        ↑V,↑↑,↑ are all escape characters

A(ppend)        B(ackup)        C(hange)        D(elete)
E(xit)          F(ind)          G(roup)         H(elp)
I(nsert)        J(am)           K(ill)          L(ist)
M(odeset)       N(otemodes)     O(utput)        P(rint)
Q(uit)          R(ead)          S(earch)        T(ype)
V(iew)          W(rite)         X(change)

$ Go to End     = Current Line  - Move Back     + Move Forward
, Context       ↑ Print Prior   / Print Current LF Print Next
← Format        ; Comment
% Call SNDMSG   ! Run program   @ Call EXEC     " Mode dialogue
```

; COMMENT

All text input from the terminal following the ";" is ignored until the occurrence of a ↑Z or a ↑Q.

1.2 POSITIONING AND SEARCHING COMMANDS

Most commands in the editor operate on the current line of text within the text buffer.

Lines in the text buffer are numbered consecutively starting at 1. The commands in this section describe ways to examine the contents of the text buffer and to change to current the line.

$ DOLLARS

$ (Dollars) makes the last line in the text buffer the *NEW* current line. It also prints the number of lines in the text buffer.

NUMBERS

Numbers which are input as commands cause the current line to become the line with the number input. If a digit is incorrectly input, it may be canceled with a ↑A, ↑H, or DEL. The entire number may be cancelled with ↑Q.

The commands ,(context), +, -, K(ill), T(ype), and M(odeset) all expect to receive numbers following. These numbers may be similarly edited with ↑A, ↑H, and DEL. ↑Q cancels the entire command. : (colon) - K(ill) and T(ype) expect to receive a count of the number of lines on which they should operate. However, if the number is preceded by a colon (:), they operate on the current line through the line number input instead of n lines starting at the current line.

Example: 12T:15. is the same as 12T4.

. (DOT)

. (Dot) is a convenient character to terminate numbers, and is a null operator to the editor. Space and carriage return would also work as well.

= (EQUALS)

= (Equals) prints the line number of the current line.

↑ (UP ARROW)

↑ (Up Arrow) moves the current line to the line preceding the current line in the text buffer and prints the *new* current line. This is the same as -/.

LF (LINEFEED)

LF (Linefeed) advances the current line one line forward in the text buffer and prints the *new* current line. This is the same as +/.

+ (PLUS)

+ (Plus) accepts a number following it, and advances the current line that number of lines forward. If no number follows, the current line is advanced one line forward. If this command is input by accident, typing ↑Q will cancel it.

- (MINUS)

- (Minus) accepts a number following it, and moves the current line that number of lines backwards. If no number follows, the current line is moved one line backwards. If this command is input by accident, typing ↑Q will cancel it.

F(IND)

F(ind) accepts a string from the terminal. It then locates the first occurrence of the string in the text starting with the current line throughout the entire text buffer. The *new* current line is the line found, or the current line if no match was found. If the search string was last found at the current line, the search will start at the *next* line (so repeated F(ind)s will locate successive occurrences of the search string in the text). If just a <CR> is typed to the prompt for a string, Find uses the string from the last F(ind) or S(earch). If the search string is a single null character (specified by control-V control-@ [↑V↑@] or control-↑ @ [↑↑@]), then XED will search for a null line (no text, only <CR><LF>). If this command is input by accident, typing ↑Q will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode. This command normally looks for the string any place in the line, independent of upper and lower case differences.

```
↑E If ↑E is specified, the match will be exact only.
↑B If ↑B is specified, the match will be only at beginning of lines.
↑S If ↑S is specified, XED will enter Change on the line(s) containing
   the string, positioning the Change cursor at the found string.
```

During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

S(EARCH)

S(earch) accepts a string from the terminal. It then locates *all* occurrences of the string in the text, starting with the current line. The *new* current line is the last occurrence found, or the starting line, if no occurrences were found. If just a <CR> is typed to the prompt for a string, Search uses the string from the last F(ind) or S(earch) command. If the search string is a single null character (specified by control-V control-@ [↑V↑@] or control-↑ @ [↑↑@]), then XED will search for a null line (no text, only <CR><LF>). If this command is input by accident, typing ↑Q will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode. This command normally looks for the string any place in the line, independent of upper and lower case differences.

↑E If ↑E is specified, the match will be exact only.
↑B If ↑B is specified, the match will be only at beginning of lines.
↑S If ↑S is specified, XED will enter Change on the line(s) containing the string, positioning the Change cursor at the found string.

During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

1.3 LINE EDITING COMMANDS

I(NSERT)

The I(nsert) command accepts text lines from the terminal. These text lines are inserted *before* the current line in the text buffer. This command is normally terminated by a ↑Z. It ↑Q is typed, any text typed on the current line is ignored and XED returns to command mode. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

A(PPEND)

The A(ppend) command accepts text lines from the terminal. These text lines are appended *after* the current line in the text buffer. This command is normally terminated by a ↑Z. It ↑Q is typed, any text typed on the current line is ignored and XED returns to command mode. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

↑I (tab) TAB APPEND

Acts like A(ppend), with a tab as the first character of the first line appended.

D(ELETE)

The D(elete) command deletes the current line. Once input there is no way to recover the line. For this reason K(ill) is considered a safer command.

K(ILL)

K(ill) accepts a number following it, and deletes that number of lines starting at the current line. The *new* current line is the line *AFTER* the deleted lines. If no number follows, only the current line is deleted. If the number following is preceded by a colon (:), instead of killing n lines, it will kill all lines from the current line up to, and including, line n.

```
Example: 12k:15.  kills lines 12,13,14,15.
```

All the lines deleted are saved in the text dump. Each subsequent kill destroys the previous text dump contents. Commands P(rint text dump) and J(am text dump) operate on the text dump. If this command is input by accident, typing ↑Q will cancel it.

P(RINT TEXT DUMP)

Each K(ill) operation saves the deleted lines in the text dump. P(rint text dump) prints the current contents of the text dump. See also J(am text dump). While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

' (SWITCH DUMP)

Exchanges the current contents of the text buffer with that of the text dump (see K(ill), J(am), P(rintdump) commands). This command is useful for performing operations with sections of large files. By K(ill)ing a section of text into the dump, then switching the dump with the buffer, operations may then be performed on the small section. Repeating the command causes repetitive switching (i.e., two ' commands in a row have no effect).

J(AM)

Each K(ill) operation saves the deleted lines in the text dump. J will jam the current contents of the text dump after the current line. The *new* current line will be the last line appended from the text dump. See also P(rint text dump).

/ (SLASH)

/ (Slash) prints the current line.

↑ (UP ARROW)

↑ (Up Arrow) moves the current line to the line preceding the current line in the text buffer and prints the *new* current line. This is the same as -/.

T(YPE)

T(ype) accepts a number following it, and prints that number of lines starting at the current line. The *new* current line is the last line printed. If no number follows, one line is printed. If the number following is preceded by a colon (:), instead of typing n lines, it will type all lines from the current line up to, and including, line n.

Example: 12t:15. types lines 12,13,14,15.

If a T command follows a T command, then the current line is advanced before the second T command, so the same line is not printed twice. If this command is input by accident, typing ↑Q will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

V(IEW)

Starting at the current line, V(iew) prints the next page of text (approximately 20 lines). The *new* current line is the last line printed. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

, (CONTEXT)

, (context) accepts a number n following, after which it types the n lines *before* the current line, the current line, and n lines *after* the current line. The effect is to type (2n+1) lines, with the current line in the middle. If n is not given, 5 is assumed. ,(context) does *not* change the current line. If this command is input by accident, typing ↑Q will cancel it.

L(IST)

L(ist) will print the entire text buffer from begining to end on the terminal. It is the same as saying TTY: to W(rite). There is a way to have a document line stating the current file name, time, and person who edited the file in the front of the file each time it is written. See) and (for further details. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

1.4 INTRALINE EDITING COMMANDS

C(HANGE)

The C(hange) command is used for editing a single line of text. There are three basic operations in the change command. First characters may be copied from the old copy into the new line (SPACE,E,S). Characters may be deleted from the old copy (D,K,R). Finally characters may be added to the new line from the terminal (I,R). C(hange) operates on the *current* line. To get a full explanation of the C(hange) subcommands input a ? while within the C(hange) command. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

Change Subcommands:

?	Prints this message.
n SPACE	Space n characters forward copying from old to new.
B	Break line - Insert current contents of new line before the current line. Change subcommands may continue with the rest of the old line, althouth Q and ↑Q cannot affect the inserted text.
n D	Delete n characters forward from the old copy.
E	Move to end of line copying from old to new.
I	Insert mode - all input inserted until next CR,LF, or ↑Z.
n K x	Delete forward until nth occurrence of 'x'.
n L x	Force alphabetics to be Lower case up to the nth occurrence of 'x'.
P	Print the rest of old line and current new line.
Q	Abort all changes since last B subcommand.
n R	Delete next n characters forward and enter insert mode.
n S x	Space forward to before the nth occurrence of 'x'.
n U x	Force alphabetics to be Upper case up to the nth occurrence of 'x'.
n V x	inVert case of all alphabetic characters up to the nth occurrence of 'x'.
CR	Copy rest of old line to new, update current line to new
LF	Update current line to new, as is
↑D	Start C(hange) over, forgetting edits so far
ESCAPE	Copy rest of old line to new, update current line to new, and re-enter C(hange) on the updated current line.

If the 'x' in K, S or V command is a CR, then it is interpreted as meaning "to the end of the line"; if it is ESCAPE, it is interpreted as "use the same character as last time."

G(ROUP)

G(roup) combines the current line with the following line to produce a single new text line. The current line is set to the new combined line. If this command is executed accidentally, using the "B" subcommand of the C(hange) command can separate the two lines.

X(CHANGE)

X(change) takes 2 strings, called OLD and NEW. It then replaces occurrences of OLD with NEW, according to the user's directions. Upon finding a line with an occurrence of OLD, it replaces all the instances of OLD with NEW, types the line on the terminal, and asks the user for confirmation. Typing a "?" to the confirmation request displays the possible options. If the user types just a <CR> to the prompt for NEW, then the occurrences of OLD will be deleted. If <CR> is typed for both OLD and NEW, Xchange uses the same strings it used the last time. If the PRINT XCHANGEs mode is set, all XCHANGEd lines

will be printed even if NOCONFIRM (*) is specified. If the search string is a single null character (specified by control-V control-@ [↑V↑@] or control-↑ @ [↑↑@]), then XED will search for a null line (no text, only <CR><LF>).

If this command is input by accident, typing ↑Q will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode. This command normally looks for the string any place in the line, independent of upper and lower case differences.

```
↑E If ↑E is specified, the match will be exact only.
↑B If ↑B is specified, the match will be only at beginning of lines.
↑F If ↑F is specified, XED will do case-sensitive XCHANGEs.
```

During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

```
X(change) Confirmation Codes
Y,<space>,CR,LF
        Accept printed changes and continue through the text buffer.
N,↑H(Backspace),↑A,DEL
        Reject printed changes and continue through the text buffer.
.       Accept printed changes and stop here.
E       Reject printed changes and stop here.
*       Accept printed changes and continue through the text buffer,
        accepting all changes without requesting confirmation.
```

If any other character is input, the confirmation request is repeated.

1.5 FILE INPUT/OUTPUT COMMANDS

R(EAD)

Read accepts a file name from the terminal and reads the contents of the file into the text buffer by appending the contents AFTER the current line. The old text buffer contents are not changed. Input and output operations to files will be encrypted if the ENCRYPT mode is set. If this command is input by accident, ↑Q or DEL will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

E(XIT)

E(xit) accepts a file name from the terminal and writes the contents of the text buffer into

this file. When the write is complete, the editor exits back to the Executive (@). If the termination is not desired, use the W(rite) command. There is a way to have a document line stating the current file name, time, and person who edited the file in the front of the file each time it is written. See) and (for further details. Input and output operations to files will be encrypted if the ENCRYPT mode is set. If this command is input by accident, ↑Q or DEL will cancel it. After this command has returned to the Executive(@), inputting CONT<CR> will continue the editor with no changes to the current line, text buffer etc.

B(ACKUP)

When a file name is entered to R(ead), W(rite), or E(xit), the editor remembers the name. When executed, B(ackup) writes the text buffer into specific versions of that file name. Normally, B(ackup) writes the buffer into version 2 of the file. If version 2 previously existed, it is renamed to version 1 before the write (the previous contents of version 1 being lost). Thus, repeated uses of the B(ackup) command write over the same files, minimizing the proliferation of new versions of the file. If the SPECIAL BACKUP mode is set, B(ackup) uses the two version numbers numerically following the highest version of the file last read or written instead of versions 2 and 1. This has the advantage that the files created by the B(ackup) are the highest versions of the file and will be referenced when the version number is defaulted (which is almost always the case). To recover the backup copy of the file after a system crash simply read as normal. Read always checks for version 2 backup copies with later write times than the requested versions. There is a way to have a document line stating the current file name, time, and person who edited the file in the front of the file each time it is written. See) and (for further details. Input and output operations to files will be encrypted if the ENCRYPT mode is set. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

W(RITE)

W(rite) accepts a file name from the terminal and writes the contents of the text buffer into this file. If the user plans to Q(uit) immediately following the W(rite), the E(xit) command might be more convenient. There is a way to have a document line stating the current file name, time, and person who edited the file in the front of the file each time it is written. See) and (for further details. Input and output operations to files will be encrypted if the ENCRYPT mode is set. If this command is input by accident, ↑Q or DEL will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

() (PARENTHESES)

Each time a file is written by B(ackup), W(rite), E(xit), or L(ist) it is possible to have an extra line added to the top of the file stating the current file name, time, and person who edited it. In addition to this information, there is also facility for a prefix string and a suffix string to be put before and after this information. These latter strings are provided to make these lines into comments for the programs that might process the files. This line will automatically be generated *only if* a prefix string has been specified.

((PREFIX)

The ((Prefix) command accepts a string from the terminal which will be used as the prefix to the automatically generated documentation line when the file is written. The documentation line is *not* added to the text buffer, but is obviously included when the file is next read into the editor. The prefix is required to trigger the automatic document line generation. Inputting (<CR> clears the prefix string and turns off the automatic document line generation. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active: ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape). Detailed explanations are available from H(elp).

) (SUFFIX)

The) (Suffix) command accepts a string from the terminal which will be used as the suffix to the automatically generated documentation line when the file is written. The documentation line is *not* added to the text buffer, but is obviously included when the file is next read into the editor. The PREFIX is required to trigger the automatic document line generation. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active: ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape). Detailed explanations are available from H(elp).

O(UTPUT)

This command writes a formatted output file suitable for printing on a hardcopy device such as a line printer (it is *not* intended for normal file writing!). It can provide titles, page numbers, and a summary of editing changes. It is affected by several modes which determine what kind of headings will appear in the output, how page numbering is to be done, and the size of the output page. These modes are found in the Format/Output mode group of the mode dialogue. When invoked, O(utput) first prompts for a heading (unless the mode affecting the heading specified not to prompt). It then asks for the name of the output file. If a <CR> or <ESCAPE> is specified for the file name, then LPT: is assumed. If this command is input by accident, typing ↑Q will cancel it. While this command is executing, typing ↑Q interrupts it and returns to XED command mode.

Q(UIT)

Q(uit) will exit the editor *without* writing a new copy of the file. After this command has returned to the Executive(@), inputting CONT<CR> will continue the editor with no changes to the current line, text buffer etc.

1.6 MODES

* (MODE DIALOGUE)

Puts the user in an interactive dialogue, allowing the user to set and examine the XED mode settings. It divides the settings into groups of modes, and allows the user to select or skip each group. Within a group, the user then may examine and optionally set any of the modes within the group. At any time during the dialogue, the user can skip the rest of the group selections and individual mode selections, and can also cancel any changes made. It also allows the user to define mode files to contain specified mode settings, and recall them later.

1.7 PROGRAM INVOKING COMMANDS

% (CALL SNDMSG)

Allows the text buffer to be sent to SNDMSG without going through EXEC or using SNDMSG's ↑B option. XED prompts for subject of message, after which user is prompted for To:, cc:, etc., just as in SNDMSG. The message will have the text buffer inserted as the body. It is important not to type ahead while the message body is being inserted, as the input will be merged randomly with the text buffer into the message body. When the message has been sent, XED will return to command mode, with the text buffer untouched. If this command is input by accident, typing ↑Q will cancel it. During the input of text for this command the following control characters are active. Detailed explanations are available from H(elp): ↑A, ↑H(backspace), ↑X, ↑W, ↑R, ↑Q, ↑V, ↑↑, ↑\, DEL and $(escape).

! (RUN PROGRAM)

Allows the user to call any TENEX subsystem or other runnable program (including the TENEX EXEC) without disturbing the state of the text buffer, current pointer, etc. XED prompts for the name of a program to run (directory defaults to <SUBSYS>, extension to .SAV) then runs the requested program. When the program is finished running (such as by QUIT to the EXEC), control returns to XED, leaving it in the same state as when the program was called. If this command is input by accident, ↑Q or DEL will cancel it.

@ (CALL EXEC)

Starts up the TENEX EXEC without disturbing the state of the text buffer, current pointer, etc. When a "QUIT" command is given to the EXEC, control returns to XED.

[(START FORK)

Starts last program run by a program-running command (!,?,@). If no program has been run, or the last fork was killed by the > command, the user is prompted for the name of a new program as in the ! command.

```
< (CONTINUE  FORK)
```

Continues last program run by a program-running command (!,?,@). If no program has been run, or the last fork was killed by the > command, the user is prompted for the name of a new program as in the ! command.

```
> (Kill fork)
```

Kills the last program run, and its fork. If no program has been run, or it has already been killed, this command has no effect.

1.8 FORMATTING

```
← (FORMAT)
```

The format command allows the user to do limited text formatting and justification on portions of (or all of) the text buffer. If the command is followed by a <CR> or <SPACE>, then the whole buffer is formatted. It can also be followed by a line range specification, analogous to the T(ype) or K(ill) commands. Thus,

```
←40 means format the next 40 lines in the buffer
    starting at the current line

←:125 means format the lines starting at the current line
      and extending to line 125
```

[Note that the formatter is an experimental facility and subject to change as its features are used, critiqued and refined.]

Several modes are relevant to the format command (set and examined by the " command). The command is enabled by the ENABLE FORMAT mode (in the Command Enable/Disable group). If justification (adding of additional spacing to even the right margin) is desired, then the Justification mode (in the Miscellaneous group) must be turned on. The margin used by the formatter is also settable by the " command (in the Miscellaneous group). Its default value is 70.

In order to protect the user from performing a format operation which produces a result the user does not expect, the lines which are being formatted are first copied to the text

dump. If the user has formatted the whole file, he can retrieve it with the ' (Switch dump) command; otherwise, he can use the J(am dump) command.

For normal operations, the formatter requires no explicit commands. Paragraphs are separated by blank lines. Each paragraph may contain "flags" which direct the formatter to perform indentation on the entire paragraph. The blank lines are retained after formatting, so that subsequent formats of the same text will have the same paragraph boundaries.

The formatter recognizes two special text lines as command directives. The first is a line with only a single, isolated tab. It directs the formatter to turn off formatting until the occurrence of another isolated tab line. This is useful to allow certain pre-formatted information (such as tables) to be skipped over by the formatter. [Note that unintended results can occur if the isolated tab lines are not properly paired!] The second command is a line with an isolated space. This acts as a paragraph break and directs the formatter to indent the following paragraph the same as the previous one. To help the user recognize the occurrence of these special command lines, XED highlights them (on terminals so equipped) or prints special identifying lines (e.g., << isolated tab >>).

1.9 XED PARAGRAPH FORMATTING

Note: This description has been formatted by the XED format command. Thus it both describes the process and demonstrates the available features.

I. Normal Paragraph Processing

The first line of a formatted block is treated as a new paragraph. A new paragraph is also created after a blank line (2 consecutive carriage-returns). The paragraph is terminated by one of the following:

- the end of the block of lines being formatted
- a blank line
- a line with a single isolated tab

In the latter case, formatting is also turned off until the next line containing an isolated tab is encountered.

A. There are two items in the XED mode file which affect the formatting process: the justification flag and the line width.

   1. The justification flag specifies whether the text should be justified to the right margin. If the flag is on, lines will be filled with as many words as can fit, then spaces will be added to even the right margin. This example is justified.

   2. The line width determines how wide the formatted lines will be. (In the case of no justification, this is the maximum length.) This example has the line width set to 70.

B. The normal paragraph processing puts the proper number of words on each line. If justification is requested, each line is also justified. The basic features of this formatting process (besides line-filling and justification) are:

   1. Paragraph indentation is not changed. If the first word is flush to the left margin, the formatted paragraph will also start flush; if the first word is indented five spaces, the formatted paragraph will also have the first word indented five spaces. However, all other spacing characters (space, tab, carriage return, line feed) within the paragraph are simply used to separate words.
      All indentation (whether just the first line of a paragraph or an entire paragraph) is done with spaces. Tabs are converted into the proper number of spaces.

   2. Hyphenated words are recognized and possibly broken. This has the side effect of possibly adding spaces (after the hyphen) in the middle of hyphenated words in later versions.

   3. All periods, colons, question marks, and exclamation points appearing at the end of words are followed by two spaces (minimum), instead of the one normally used between words.

II. Flagged Paragraph Processing

In addition to normal paragraph processing, the facility also provides for paragraphs which are entirely indented. These are called "flagged" paragraphs.

A. The general form of a flagged (indented) paragraph is:

1. A flagged paragraph starts with at least one spacing character (tab or space).

2. This is followed by a flag. The flags are described in section III below.

3. Finally, the flag must be followed by at least one spacing character.

* To reiterate, in order for a flag to be recognized, it must be the first word in a paragraph with spacing both before and after. This is done to prevent normal paragraphs from being mistaken for flagged paragraphs. Following are the differences between the processing of flagged paragraphs and normal paragraphs.

B. Flag Formatting:
The flag will be indented in the formatted result the same amount it was indented in the original text. This is similar to the initial paragraph indenting provided for normal paragraphs. The flag will be followed by two spaces which will not be available for expansion during justification.

C. Succeeding Lines in Flagged Paragraphs:
All lines of the flagged paragraph will line up with the first line of the paragraph and will be indented from both margins.
INDENTED PARAGRAPHS WITHOUT FLAGS
When you want more than one paragraph to line up under a flag, insert a line with a single isolated space between the two paragraphs. The next paragraph will line up automatically with the previous one. This section makes extensive use of this facility.

III. Description of Flags

Flags are used to indent entire paragraphs. There are four kinds:

SINGLES: A single is one of the following: * (star), + (plus), - (hyphen), % (percent), and o (lower-case letter O).

SECTIONS: A section flag can be a single letter, a letter and a number (0-9 only), any number 0-99, or a number (0-9) and a letter. It cannot be two letters since a paragraph could start with Mr. Smith or Dr. Jones. Examples: A,1,A1,22,1a. A section cannot be more than 2 characters.

ROMANS: Any roman numeral between 1 (I) and 399 (CCCXCIX). The roman numerals can be upper- or lower-case.

NAMES: Any string containing only alpha-numerics, . (periods) or - (dashes).

Four classes of flags are recognized by the following punctuation:

A. Any SINGLE by itself (spacing before and after) is considered to be a flag.

B. Any SINGLE, SECTION, or ROMAN followed by a . (period) or a) (close parenthesis) is considered to be a flag.

C. Any SINGLE or SECTION enclosed in balanced brackets is considered to be a flag. Recognized brackets are: <>, , [], and ().

D. Any NAME followed by a colon (:) is also considered to be a flag.

EXAMPLE FLAGS

Note: [This section is not formatted. The following line contains a single isolated tab which turns off the formatting process.]

Example flags with appropriate punctuation:

Selected from:	SINGLE	SECTION	ROMAN	NAME
Flag group A: SINGLE ALONE	*			
	+			
	-			
	o			

```
                    %

Flag group B:       (*)     [A]
BRACKETED           <o>     (D3)
                    {+}     <7>

Flag group C:       o)      23.     IV)
SECTION DESIGNATOR  *.      2a)     xxxi.
                    +)      Q.      CX)

Flag group D:                               Flagword:
KEYWORD                                     Case-176:
                                            STATUS.LAST:
```

1.10 CONTROL CHARACTERS

Control characters are input by depressing the CONTROL key (which is a shift key and by itself does not transmit any code to the computer) followed by the appropriate character. Control characters are represented as an ↑ preceding the character. E. G.: ↑A represents Control A.

1.11 INTERRUPT CONTROL CHARACTERS

These characters may be input at any time. ↑C and ↑T are available anywhere in TENEX, while the others are only supported by XED.

↑C (CONTROL C)

This stops the editor and goes immediately to the Executive(@). The editor may be continued by inputting CONT<CR>.

↑T (CONTROL T)

This will print the current load average of the system.

↑N (CONTROL N)

This will print the current line number. This is useful for tracking the progress of R(ead), W(rite), B(ackup), E(xit), F(ind), and S(earch).

1.12 TEXT INPUT CONTROL CHARACTERS

These characters are available whenever the user is inputting a text string from the terminal.

↑A (CONTROL A)

↑A, ↑H(backspace), DEL(rubout), will all delete the last character input. DEL(rubout) additionally aborts commands which accept file names. In this file name situation DEL is active during R(ead), W(rite), E(xit), H<CR> [Manual Request], and when B(ackup) requests a file name because not file name has yet been specified. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑W (CONTROL W)

↑W deletes the last word input. Words are delimited by tabs and spaces. ↑W also deletes and spaces or tabs which might have followed the last word input. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑X (CONTROL X)

↑X deletes the current line being input from the terminal. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑R (CONTROL R)

↑R retypes the current line being input. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑Q (CONTROL Q)

This will stop an executing command. It can stop B(ackup), F(ind), L(ist), O(utput), P(rint text dump), R(ead), S(earch), T(ype), V(iew), W(rite) and X(change). ↑Q is also active whenever a command or text is being input. If typed while entering the number parameter to a command like K(ill) or T(ype), it will abort the command. If typed while Inserting or Appending, the current text line will be discarded and XED will go back to command mode.

↑V (CONTROL V)

↑V "quotes" the next character input. It is used to input characters which would normally cause control or editing functions (e.g., ↑Z, ↑X, ↑Q, ↑B). For example, to put "ABC↑Z" in a

text line, one would enter ABC↑V↑Z. ↑V would be entered into text by ↑V↑V. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑↑ (UP ARROW UP ARROW)

↑↑ (Up Arrow Up Arrow) "control shifts" the next character input. (e.g. ↑↑C inputs ↑C, while ↑↑c inputs ^). This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

↑ (UP ARROW BACK SLASH)

↑\ (Up Arrow Back Slash) "case shifts" then next character input. (e.g., ↑\C inputs c, ↑\c inputs C, while ↑\^ inputs ↑C). This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

1.13 SEARCH MODIFICATION CONTROL CHARACTERS

These characters cause special actions to be performed during the execution of certain commands. To enter them as text for these commands, they must be "quoted" by ↑V or ↑↑.

↑B (CONTROL B)

↑B specifies that F(ind), S(earch), and X(change) must match only at the *beginning* of a text line. (The default is to match anywhere in the text line.) The ↑B may be typed either before the command it affects (F,S or X), or during the typing of the string affected. If this command is entered more than once, it reverses the effect of the previous one (or the prevailing default setting), i.e., it acts as a toggle.

↑E (CONTROL E)

↑E specifies that Find, Search, and Xchange must Exactly match the text string, including case. (The default is to match exclusive of case.) The ↑E may be typed either before the command it affects (F,S or X), or during the typing of the string affected. If this command is entered more than once, it reverses the effect of the previous one (or the prevailing default setting), i.e., it acts as a toggle.

↑F (CONTROL F)

This command affects the X(change) command. When specified, it causes XED to do case-sensitive XCHANGEing. In this mode, XED categorizes the found string into one of four classes (UPPER CASE, lower case, Capitalized, or aRbitrAry). When it substitutes the

replacement string, it maps it into the same class as the found string (if the found string is arbitrary, XED makes no change). The ↑F may be typed either before the X(change) command, or during the typing of the OLD string. If this command is entered more than once, it reverses the effect of the previous one (or the prevailing default setting), i.e., it acts as a toggle.

↑S (CONTROL S)

This command affects the searching commands (F,S). When specified, it causes XED to enter Change for the line within which the string was found, positioning the Change cursor at the found string. The ↑S may be typed either before the command it affects (F,S), or during the typing of the string affected. A mode in the Miscellaneous mode group allows the user to specify that the C(hange) cursor be positioned at the end of the found string. Normally, it is positioned at the beginning. If this command is entered more than once, it reverses the effect of the previous one (or the prevailing default setting), i.e., it acts as a toggle.

$ (ESCAPE TO CHANGE)

During the insertion of text lines, the escape character passes the line currently being input to C(hange). When C(hange) is terminated in the standard way, the line input process continues. If escape is typed when the text is empty, the text line will first be initialized to the *last* string entered in the current context, e.g., if the string is being entered for a F(ind) command, the string used in the last F(ind)/S(earch) command will be used; if the string is being entered into the text buffer by an A(ppend), the last string entered during the last A(ppend)/I(nsert) command is used. This control character is active during A(ppend), I(nsert), C(hange), F(ind), S(earch), X(change), ((set prefix),) (set suffix), and % (SNDMSG).

2. INTRODUCTION TO TECO

TECO is one of the most powerful and concise text editors in existence. As with most text editors, the basic idea is to

1. **Bring a file into the "main text buffer" (with ;Y - see below),**
2. **Edit the contents of the buffer, and**
3. **Save the buffer as a latter version of the same file (with ;U).**

We will use the following symbols in what follows:

```
 $ = escape = altmode
CR = return
↑D = control D; will be echoed as $ just like escape.
     Used at the end of search or replacement text.
```

2.1 STARTING TECO

```
Type @TECO CR to the EXEC.
```

TECO types a star (*) when it is ready for a command. TECO commands end with *Escape* -- not *Return*. TECO does nothing until the final escape is typed.

Several TECO commands may be strung together (with nothing between them or a space if desired). TECO executes all the commands when the final escape ($) is typed. Note that this is different than the EXEC, where each command is terminated by return.

2.2 TECO COMMAND ERRORS

If you try to execute an erroneous command, TECO will type a question mark followed by an error number, followed by a few characters of the command in the vinicity of the error, and then a star to await another command. The error numbers may be looked up in a book (the DEC TECO manual), but the error is almost always obvious (after some experience). A usually good response to an error is to execute the -2T2T command (see below) to determine if any harm was done by the bad command and to find out where the pointer is, and then to re-execute the intended correct command.

2.3 FIXING TYPING MISTAKES

```
↑A  deletes the last character typed (may be used repeatedly).
    Echoes as a backslash followed by the deleted character.

↑Q  deletes the current line (may be used repeatedly).

↑R  retypes the current line.  Use after several control A's have
    made the line unreadable or after control Q to see where you
    are.
```

TWO RUBOUTS (labelled DEL on some terminals) deletes the current command and gets back to the TECO *. USE IF YOU GET THE FILE NAME WRONG FOR ;Y OR ;U.

2.4 CONTROL-C ACCIDENTS

If Control-C is accidentally (or purposely) typed, TECO will be interrupted and the user will be back talking to the EXEC (with the '@' sign). TO GET BACK INTO TECO after Control-C with no ill effect, type the EXEC command CON (short for CONTINUE) followed by return. This will put the user back in TECO as if no ↑C had been typed, except that no TECO star will appear and a few characters may be lost if the user was in the middle of an insertion. To get the star (*), type two rubouts (labelled DEL on some terminals), and then proceed with normal TECO commands.

2.5 FILE COMMANDS

```
GET a File:        *;YS
                   INPUT FILE: XS [CONFIRM]S

SAVE a File:       *;US
                   OUTPUT FILE: X CR [NEW FILE]S     with X return or -
                   OUTPUT FILE: XS [NEW VERSION]S    with X escape
```

Unyanks the buffer to file X and *wipes out the buffer*. If X is a *new name*, use return after X. If X is just a *new version*, use escape after x and the system will supply the new version number.

2.6 TEXT EDITING COMMANDS

(Remember - several commands may be strung together, but TECO will do nothing until the final escape tells it to execute the commands.)

Many of the commands work with respect to the *pointer* into the *main text buffer* (buffer for short). The buffer is where all the insertions, deletions, replacements, etc. go on, and the pointer is a moveable position marker pointing into the buffer. It is always positioned *between* two characters (e.g., between the last character of one line and the first character of the next), or before the first character in the buffer, or after the last.

2.7 MOVING THE POINTER

```
BJ        Jump pointer to BEGINNING of the buffer.

ZJ        Jump pointer to the END of the buffer.

nL        Move pointer n lines.
```

Toward last line if n is plus, toward first line if n is minus. 0L moves pointer to beginning of the current line if it was in the middle of the line (as after a search or replace command). If n is absent 1 is assumed. Examples: L -3L 5L.

2.8 TYPING OUT TEXT

```
nT        Type out n lines.  ===> DOES NOT MOVE POINTER <===
```

Actually types out from current position of pointer to end of next n lines, so if pointer is in the middle of a line, type-out will start in the middle of the line too. If n is plus, types following lines. If n is minus, types preceding lines. if n is absent, 1 is assumed. examples: T -3T 5T 0T (types from beginning of current line to pointer if pointer is not at beginning).

Note that the T command is the only way to get TECO to type out the text in the buffer. None of the other commands, like L for moving the pointer, cause anything to type out on the terminal.

```
HT        A special form of the T command.
```

Types out the entire buffer, regardless of where the pointer is (still doesn't move the pointer). To stop the typeout before the end, use two rubouts. A reasonable way to type the buffer slowly is to repeatedly execute the commands 16L18T (after starting with BJ18T). This will move the pointer 16 lines and then type the next 18, which provides a two-line overlap for context.

2.9 DELETING LINES

```
nK          Kills (deletes) n lines.
```

See T command for discussion of n. Note that HK will delete all the text in the buffer, leaving it empty.

2.10 INSERTING TEXT

```
Itext↑D     Insert text starting at current position of pointer.
```

The text may contain returns to insert several lines, *but no return is necessary after the I.* After the insertion, the pointer will be at the *end* of the inserted text. The text may end with ↑D if more commands are to be typed, or with $ (escape) if the insertion is to be done immediately.

```
;G          EMERGENCY
```

If the I is forgotten at the beginning of a long insertion, TECO will not know what to do with the text which was to be inserted and will give some error message.

```
;G          LIFESAVER
```

If the user types types ;G escape, TECO will take the last command typed which was longer than 16 characters, and insert it into the buffer at the current position. Thus, if the I of a long insert command is forgotten, ;G will insert the text that was to be inserted.

2.11 SEARCHING FOR TEXT TO MOVE POINTER

```
Stext↑D   Search for "text".
```

Pointer will be positioned at the end of the text found. Thus, SABC↑D0LT searches for "ABC", moves the pointer to the beginning the that line (the 0L), and types out that line (the T). If the Search command is preceded by minus (-), then the search will be backwards from the current position of the pointer (but the pointer will still be left at the same end of the found text as for a forward search).

2.12 REPLACING TEXT

```
REPLACE Command      nRold↑Dnew↑D0LT
```

Replace the next n occurrences of "old" text by "new" text. If n is omitted, 1 is assumed. The 0LT is not necessary but is recommended to move the pointer to the beginning of the line and type out the line to show the replacement done. The old and new text may both contain carriage returns.

2.13 REPEATING COMMANDS

```
n<commands>  Repeat the commands within the brackets n times.
```

If n is omitted, *infinity* is assumed. See example 2.

EXAMPLES: (spaces are used between commands to make the examples easier to read; they are not necessary)

1. `*BJ INEW LINE 1↑D BJ 2T $`

`BJ`	jump pointer to beginning of buffer.
`I...↑D`	insert new first line.
`BJ`	jump back to (new) beginning.
`2T`	type two lines.
`$`	escape - execute the above commands.

Nothing is done until the escape is typed.

2. `*BJ <ROLD↑DNEW↑D 0TT>`

Starting at the beginning of the buffer (the BJ), keep searching for "old" and replace each occurrence by "n." Each time the replacement is done, type out that line (the 0TT). The process will stop when no more "old"'s exist and will retype the last line in which a replacement was done. It is the brackets <> which cause the enclosed commands to be done over and over.

3. The following is a complete TECO example of creating and modifying a file. If studied carefully it should answer many questions. User type-in is all upper case (except text), system response is in lower case.

```
@TECO CR
*IA garbage collection takes approximately 1 sedon on the
IBM 7090 computer.  It can be recognized by the stationary pattern
of the MQ lights.
$
*;U$
output file: DEMO CR [new file] $ ↑[
*↑C
@TECO CR     (actually the ↑C above and this line are not necessary)
*;Y$
input file: DE$mo.;1 [confirm] $ ↑[141 chars
*R sedon ↑D second ↑D 0LT $
A grabage collection takes approximately 1 second on the
*ZJI     Any trap which prevents completion of a garbage collection will
create a panic condition in memory from which there is no recovery.
↑D HT $
A GARBAGE COLLECTION TAKES APPROXIMATELY 1 SECOND ON THE
IBM 7090 computer.  It can be recognized by the stationary pattern
of the MQ lights.
     Any trap that prevents completion of garbage collection will
create a panic condition in memory from which there is no recovery.
*;U$
output file: $ demo.;2 [new version] $ ↑[
*↑C
@
```

The commands above form a basic subset that will get the new user started. The following are additio..al commands that will be useful after the new user has some experience.

2.14 INSERTING TEXT IN THE MIDDLE OF A LINE

Assume we have the following line, and we wish to put "word" between the "a" and the "missing":

```
This is the line with a missing.
```

Either command will do it and type the corrected line:

```
*Ra m ↑D a word m D 0LT $
```

or

```
*Sa ↑D Iword ↑D 0LT $
```

2.15 DELETING CHARACTERS

This may be done with a replace command with nothing between the two control Ds, or with the nD command. nD deletes n characters after the pointer if D is positive, or n characters before the pointer if n is negative.

Assume the pointer is at the start of the line:

```
0LTThis line has a mistake in it.
```

and we wish to get rid of the "0LT". Either command below will do it and type the new corrected line:

```
*R0LT ↑D ↑D 0LT $
```

or

```
*3DT $
```

2.16 MOVING LINES VIA Q-REGISTERS

TECO has thirty-six "Q registers" named A through Z and 0 through 9. They are useful for many things, but we now examine only two Q register commands which can be used to move lines of text from one place in the buffer to another.

Pick Up Lines: nXi

Store in Q register i all characters from the current position of the pointer through the nth line from the pointer. *This removes the text from the buffer.* Thus, if the pointer is at the beginning of a line, then 4XA will pick up that line and the next three, and place the four lines into Q register A.

Put Down Lines: Gi

Place the text in Q register i into the buffer at the current position of the pointer. After the insert, the pointer will be at the end of the inserted text.

Example: assume the buffer contains the following:

```
LINE A
LINE B
LINE C
LINE D
```

Then the command 0J 2L 2XA -1L GA will move lines C and D to between lines A and B as follows:

```
0J    move pointer to beginning of the buffer.
2L    move pointer to beginning of LINE C.
2XA   pick up the current line and the next line (LINE C and  LINE
      D)  and  place  them  into Q register A, leaving the pointer
      after LINE B at what is now the end of the buffer.
-1L   move the pointer to the beginning of LINE B.
GA    insert the contents of Q register A at the current  position
      of the pointer.
```

Final result:

```
LINE A
LINE C
LINE D
LINE B
```

with the pointer at the beginning the the last line: LINE B.

2.17 MOVING TEXT THAT DOES NOT START AND/OR END ON A LINE

Assume we have something like:

...end of text to stay. Meantime, the world in which we exist has other aims. But it will pass away, burned up in the fire of its own hot passions; and from its ashes will spring a new and younger world, full of fresh hope, with the light of morning in its eyes. Start of text to stay...

To move all but the first and last sentences we use:

```
*Send of text to stay.$      Put pointer at the start of "Meantime,".

*.U1 $                       Put pointer position into Q register 1.
                               ("." means current pointer position)

*Sin its eyes.$              Put pointer at the end of "eyes.".

*Q1,.X1 $                    Pick up the text from "Meantime,"
                             through "eyes." and put it in Q
                             register 1.  The full X command is:
                             first-char-pos , last-char-pos X q-reg

*S to where ever we want to put the text $

*G1 $                        Put text in Q register 1 at the current
                               position of the pointer.
```

This technique can also be used to move large blocks of text without counting the number of lines in a block even if the text does start and end on line boundaries.

3. DCOPY - DIABLO PRINTER TERMINAL PROGRAM

The DCOPY program was created at ISI to handle the production of output on the DIABLO Printer Terminal; it is available on the ISI-TENEX and ISI-TOPS-20 operating systems.

3.1 DIABLO PRINTER TERMINAL

The DIABLO is a high quality hard copy terminal that produces finished output on either regular (letterhead or bond) or computer paper (fan-folded or continuous roll) in a variety of pinwheel type fonts, boldface characters, and two-color (red and/or black) type. While the DIABLO can be used as a standard terminal to create, edit and format files using the text handling subsystems supported by TENEX and TOPS-20, its most effective use is as a "personal XGP": the user logs in, runs DCOPY (a program developed for DIABLO), completes the task, then logs off.

3.2 DCOPY PROGRAM

DCOPY operates much like the XGP, i.e., it accepts either a formatted file directly for output or a file for processing with DCOPY commands. This document will present three scenarios describing DCOPY: first, the processing of a formatted file; second, the processing of a file using DCOPY commands; and third, a brief description of the DIABLO special features. It is assumed that the user is familiar with text editors and formatting programs and will have reviewed the <DOCUMENTATION> DCOPY.MANUAL. Included in this document is a guide to formatting text files for output on the DIABLO.

The key to successful and nonfrustrating use of the DCOPY program is the formfeed character (↑L) which, when encountered at any point in a file, causes DCOPY to pause and sound a buzzer-like (or bell) sound. At a pause/bell the user may either insert the next page of a multipage letterhead or bond paper output file, or change type fonts by inserting the font pinwheel (much like changing elements in a Selectric typewriter). Typing a space as a signal to DCOPY terminates the pause/bell and DCOPY continues typing of output. (The space will not be inserted at the terminal.)

CONTROL C (↑C) terminates DCOPY and returns the user to TENEX Exec. Should the user wish to abort output at the pause/bell position, the space terminating the pause/bell must be typed first, then the CONTROL C. The pause/bell position in *DCOPY expects the*

user to type a space and does not recognize a Control C until the space has been typed. The use of the continue command at TENEX Exec is, of course, not desirable in running DCOPY. Users running DCOPY via TELNET must set transparent.mode in TELNET.

3.3 FILES FORMATTED IN XOFF OR RUNOFF

It is important to remember that both the first and last lines of XOFF and RUNOFF output files are formfeeds (↑L); in processing output from XOFF and RUNOFF the first DCOPY pause/bell should be terminated and the second should be the point at which the paper is changed or positioned. At the end of a XOFF or RUNOFF file the first pause/bell is the last line of the file (formfeed). The paper can be removed or repositioned, but terminating the pause/bell does not end the session. One more must be terminated before DCOPY exits.

3.4 XED FILES

In XED O(utput) files the last line of the file is a formfeed,which again requires two pause/bell/space operations to terminate DCOPY. XED files using only the Format (←) command should have formfeed characters (↑L) inserted in the file at desired page breaks.

3.5 NLS FILES

NLS OUTPUT TERMINAL FILES do not begin with a formfeed (↑L), but print the dashed line, then pause and sound the bell. It is recommended that the dashed line be removed by including the Output Processor Directive ".NUMDASH-0;" in the origin statement. Using the OUTPUT TERMINAL OK command outputs the file directly to the terminal and does not require running the DCOPY program. Here the DIABLO is being used as a standard terminal; the special features utilized by DCOPY cannot be used. The OUTPUT QUICKPRINT FILE command output should use the OUTPUT QUICKPRINT NO (headers) FILE COMMAND to remove everything from the header but the page number. Viewspecs are observed but not Output Processor Directives. Use this file as input to the DCOPY program. The OUTPUT SEQUENTIAL FILE command creates an unformatted sequential file which can be formatted with XOFF or RUNOFF.

3.6 EXAMPLES OF DCOPY USE

SCENARIO I: The user has created and formatted a file for output to the DIABLO. Standard login procedures are observed. The user can expect to receive standard login messages, i.e., downtime information, notice of new mail, etc. Since certainly this information is not part of the file to be processed, nor is calling the DCOPY program, inputting the file, and the program request to position the paper, this operation should be done on scrap paper and discarded, as should DCOPY's notification of "DONE" at the completion of the program.

```
@log <username> <password> <account>
 Job # on TTY # Day Month Year Time

@dc(ESC)OPY.SAV;202

INPUT FILE: <filename> [Confirm]

Position paper, then type a space.
```

At this point the user has the option to either position the fan-fold paper in the DIABLO, or insert letterhead or bond paper. As previously mentioned, XOFF and RUNOFF output files begin and end with formfeeds (↑L) and the user will of course type a space to terminate the the pause/bell. When the second pause/bell occurs, the user should then position the paper and type a space to terminate. DCOPY then proceeds to output the file. When DCOPY encounters a formfeed (↑L) in the file it will pause and sound the bell, which allows the user to insert another sheet of paper for the next page of the document. If the user is outputting the file on continuous-form paper, he repositions the paper to the top of the next page in the file and types a space to continue printing the output. When the file has been completely typed, DCOPY will pause and sound the bell, which allows the user to remove the typed output, replace the standard paper or reposition continuous-form paper, and type a space. DCOPY terminates the session and exits with:

```
DONE.
@
```

The user is now at TENEX Exec and, if dissatisfied with the output, can re-edit the file and re-process it with DCOPY, create additional copies by rerunning DCOPY, or simply logout from the DIABLO.

SCENARIO II: The user has created a file for DIABLO output and wishes to examine and use DCOPY commands. It is assumed that the user will have reviewed the DCOPY commands in <DOCUMENTATION>DCOPY.MANUAL. In this mode the user confirms the filename with a comma, DCOPY goes into command mode and signals the user by typing two asterisks (**) that it is waiting for a command. Typing ? at command mode (**)

lists possible commands. The user responds to commands either by confirming them with a carriage return or (where = (equals) appears) in commands by providing an argument and carriage return, i.e.,

```
** DOUBLE SPACE (carriage return)
** LINES/PAGE = 54 (carriage return).
```

Command mode in DCOPY terminates when the user types his last command and two carriage returns. The user is now ready to process his file. In this example the DCOPY commands are merely listed for purposes of illustration.

```
@log <username> <password> <account>
 Job # on TTY # Day Month Year Time

@dc(ESC)OPY.SAV;202

INPUT FILE: <filename> [Confirm],

** ? ONE OR ANY LOGICAL COMBINATION OF THE FOLLOWING:

10 CHARS/INCH
12 CHARS/INCH
6 LINES/INCH
8 LINES/INCH
CHARS/INCH =
DOUBLE SPACE
LINES/INCH =
LINES/PAGE =
PAUSE AT FORM FEEDS
PRINT FORM FEEDS
SINGLE SPACE
TEN STANDARD
TWELVE STANDARD

** d(ESC)OUBLE SPACE (carriage return)

** (carriage return)

Position Paper, Then type a space.
```

At this point the user continues as described in Scenario I.

SCENARIO III: DIABLO Special Features are boldface type (overprinting), two-color (red/black) typing, and multiple fonts. While underlining is not a "special feature," we will treat it as such and describe the technique for using it in DIABLO output.

3.6.1 Underlining

Use standard XOFF underline commands when creating file. Process through XOFF for output formatted for the DIABLO and specify "B" (to indicate backspacing) for underline option. Run the formatted file under DCOPY as described in Scenario I.

3.6.2 Boldface Type

Use standard XOFF underline commands when creating a file for XOFF output for DIABLO, typing O (to indicate overprinting) for the underline option. Run the file under DCOPY as described in Scenario I.

Combinations of special features, or using more than one special feature at a time, requires re-editing of the XOFF output file to insert the special characters telling DCOPY to produce the special feature desired. Care must be taken in editing an XOFF output file, since some formatting characters in the output file are not visible to the user. Once the XOFF output is re-edited for special feature use, it *cannot* be reprocessed through XOFF.

3.6.3 Underlining and Boldface Type

Process file through XOFF for underlining (as described above), then edit the XOFF output by inserting ↑H and the character to be printed in boldface immediately after the ↑H, i.e., if the word "boldface" were to be printed in boldface type they would appear as b↑Hbo↑Hol↑Hld↑Hdf↑Hfa↑Hac↑Hce↑He in the re-edited file. The ↑H is a backspace character and the letter following it will print in the same place as the letter preceding it. The file is then run under DCOPY as described in Scenario I.

3.6.4 Two-Color (Red/Black) Type

Process file through formatting program and re-edit output. Black ribbon printing is the DCOPY default. At each place in the file where red type is to appear, insert ESCAPE (or ALTMODE)A at the beginning of the word or phrase to be typed in red and ESCAPE (or

ALTMODE)B at the end of the word or phrase to be printed in red type. If the phrase "typed in red" were to be actually typed in red, writing the ESCAPE (or ALTMODE) as a "$", the phrase would look like this: $Atyped in red$B. Output the file under DCOPY as described in Scenario 1.

3.6.5 Multi-Font Output

Re-edit the XOFF file for DIABLO output by inserting a Control L (↑L) at each place in the file that you wish the DIABLO to pause to allow font change. (It is assumed that the user has reviewed the DIABLO manual and is familiar with font-changing methods.) Proceed to output the file under DCOPY as described in Scenario 1.

3.7 FORMATTING TEXT FILES FOR THE DIABLO PRINTER TERMINAL

Three or four basic steps are required for outputting formatted text files on the DIABLO (the Xerox 1700 or the Xerox 3010 Terminal). Briefly, the steps are:

1. Creating the file
2. Formatting the file (adding page breaks, paragraph justification, etc.)
3. Adding type font or ribbon color change commands to the file
4. Copying the file to the output terminal

This document concentrates on steps 1 and 2 for preparing a file to be used with the DCOPY program. Steps 3 and 4 are described in detail in the <DOCUMENTATION>DCOPY.MANUAL.

There are three common text editors used for creating files on the TENEX or TOPS-20 operating systems: NLS, XED, and TECO. The method used for formatting files is dependent on which text editor is used; some editors contain formatting tools. For example, XED has the "←(format)" and "O(utput)" commands, and NLS has the Output Processor. Other editors rely on separate text formatting programs to add page breaks, justify paragraphs, etc. For example, TECO can be used to create a file with commands for formatting with XOFF.

If NLS is used to create the file, there are four options for formatting the file for the DIABLO. Of these four options only the first two allow the use of the Output Processor Directives. The third puts out a simple default format with an optional header; the fourth

puts out an unformatted text file which can be formatted with XOFF. (This is not recommended, as you must at least put a ".verbatim" command as statement one starting in column one, and any other XOFF commands would be difficult to insert in the file.)

3.8 USING THE OUTPUT TERMINAL FILE COMMAND

The OUTPUT TERMINAL FILE command will output a formatted sequential file considering Output Processor Directives. If no Output Processor Directives are specified, the file will be paginated with the page number at the bottom of the page and a dashed line at the page break. To remove the dashed line, include the Output Processor Directive ".NUMDASH=0;" in your origin statement. The OUTPUT TERMINAL FILE command will ask three questions before processing the file: "Send Form Feeds?", "Wait (at) page breaks?", and "Go?". Answer "Yes" to "Send Form Feeds?" as the DCOPY program will expect form feed characters. Answer "No" to "Wait (at) page breaks?" as you are outputting to a file, not the terminal. Answer "Yes" to "Go?" as you are outputting to a file and don't have to position your paper. Use this file as input to the DCOPY program.

3.9 USING THE OUTPUT TERMINAL OK COMMAND

The OUTPUT TERMINAL OK command formats and copies a file to the terminal in one step. As with the OUTPUT TERMINAL FILE command, Output Processor Directives will be considered, and the file will be paginated with page numbers at the bottom of the page and a dashed line at the page break. To remove the dashed line, include the Output Procesor Directive ".NUMDASH=0;" in your origin statement. You will also be asked the same questions as for the OUTPUT TERMINAL FILE command. Answer "Yes" for "Send Form Feeds?". Answer "Yes" for "Wait (at) page breaks?" as this is when you will change your paper (you type a carriage return to proceed when it is changed). Answer "Yes" to "Go?" when you have your paper positioned properly. While this option does not require running the DCOPY program, the DIABLO is now treated like any other terminal and the special features used by the DCOPY program (e.g., printing backwards) cannot be used.

3.10 USING THE OUTPUT QUICKPRINT FILE COMMAND

The OUTPUT QUICKPRINT FILE command formats a file with pagination and an optional header which includes the date, file name, and page number. To remove everything from the header but the page number, use the OUTPUT QUICKPRINT NO

(headers) FILE command. Viewspecs are observed but not Output Processor Directives. Use this file as input to the DCOPY program.

3.11 USING THE OUTPUT SEQUENTIAL FILE COMMAND

The OUTPUT SEQUENTIAL FILE command puts out an unformatted sequential file, which can then be formatted with XOFF, RUNOFF, or some other text formatting program. For XOFF this requires at least a ".verbatim" command as statement one starting in column one. Any further XOFF commands would be difficult to use if any further NLS editing were required.

Use this file as input to the text formatting program. If XOFF is used, confirm the output file specification with a comma to format the file for a device other than the XGP. You will be asked five questions which you can answer as desired. The following answers are suggested for the DIABLO. See the XOFF Manual for further information.

```
Do you want the file formatted for the XGP (Y or N)?  N
UNDERLINE (L,C,B,O,or N)?  B
SWITCHES:  carriage return
SIMULATE FORM FEED (Y or N)?   N
PAUSE?  N
```

Use this file as input to the DCOPY program.

If XED is used to create the file, there are two options for formatting the file for the DIABLO. The first option formats the file with the XED "←(format)" and "O(utput)" commands. The second option puts out an unformatted file which can be formatted with XOFF, RUNOFF, or some other text formatting program.

3.12 USING The ←(FORMAT)/O(UTPUT) COMMAND

The "←(format)" command formats portions of (or all of) the text buffer. You separate paragraphs by blank lines, and you can intermittently turn formatting on and off by separating tables and diagrams which you do not want formatted by lines with an isolated tab. In order to use the "←(format)" and "O(utput)" commands, you must set up an alternate mode file by using the mode dialogue command, a double quote. You will be asked several questions about how you want XED to work for you. You can tailor your formatting by answering as desired for the subsequent Format/Output mode questions. When you are asked if you want to make these changes permanent, answer "Yes," name the file

(something like DIABLO.MODES), and save this file so you can read it back in later using the double quote command. The following is a good example of how you might set up your format modes for using letterhead paper on the DIABLO.

```
Enable ←(format) command -- Yes
Enable O(utput) command -- Yes
Setting of Format/Output line width is 61
Justify formatted text -- Yes
Setting of Number of lines in output page is 54
Text Header -- Yes
Output type marks -- No
Page sub-numbering -- No
Number page 1 -- No
Page numbers at bottom -- No
```

After you have formatted your text buffer, you can copy it to a file by using the "O(utput)" command. You will be asked to specify a text header to be placed on each page except page one. If you type only a carriage return you will get only the page number. If you do not have the Text Header option set, you will get the file name in the header. See the *XED REFERENCE MANUAL* for further information. Use this file as input to the DCOPY program.

3.13 USING THE W(RITE) COMMAND

The W(rite) command copies the text buffer without formatting it to a file. This file can then be formatted with XOFF, RUNOFF, or some other text formatting program. Use this file as input to the text formatting program. If XOFF is used, confirm the output file specification with a comma to format the file for a device other than the XGP. You will be asked five questions which you can answer as desired. The following answers are suggested for the DIABLO. See the XOFF Manual for further information.

```
Do you want the file formatted for the XGP (Y or N)?  N
UNDERLINE (L,C,B,O,or N)?  B
SWITCHES:  carriage return
SIMULATE FORM FEED (Y or N)?   N
PAUSE?  N
```

Use this file as input to the DCOPY program.

3.14 USING THE ;S COMMAND IN TECO

If TECO is used to create the file, there is no way to format the file within the editor unless you have enough expertise to write your own TECO macro. You need to output an unformatted file and format your text with XOFF, RUNOFF, or some other text formatting program.

The ";S" command copies the text buffer without formatting it to a file. This file can then be formatted with XOFF, RUNOFF, or some other text formatting program.

Use this file as input to the text formatting program. If XOFF is used, confirm the output file specification with a comma to format the file for a device other than the XGP. You will be asked five questions which you can answer as desired. The following answers are suggested for the DIABLO. See the XOFF Manual for further information.

```
Do you want the file formatted for the XGP (Y or N)?  N
UNDERLINE (L,C,B,O,or N)?  B
SWITCHES:  carriage return
SIMULATE FORM FEED (Y or N)?   N
PAUSE?  N
```

Use this file as input to the DCOPY program.

4. INTRODUCTION TO XOFF

This document is intended to be an introduction for the novice user to the procedure for generating documents to be output on the XGP. It assumes that the reader is familiar with RUNOFF and its workings as well as the general surface structure of the XGP system environment. The full XOFF reference manual can be found in <DOCUMENTATION>XOFF.MANUAL and information about changes made to the XOFF program is available in <DOCUMENTATION>XOFF.CHANGES.

At the simplest level, XOFF is compatible with RUNOFF (except for the RUNOFF .PAGE SIZE command, which should be deleted; its functions are taken over by the XOFF .PAPER SIZE and .PAPER HEIGHT commands, both discussed below). You can take a file which you would have input to RUNOFF and instead give it to XOFF. The best way to become familiar with the XGP and its environment is to do exactly that: give a file formatted for input to RUNOFF to XOFF, then send that file to the XGP by using the XGP program, which is described in a separate documents found in <DOCUMENTATION>XGP.DOC and <DOCUMENTATION>XGP.CHANGES.

4

XOFF will allow you to format text for a terminal, for the line printer or for the XGP. XOFF is a regular TENEX and TOPS-20 subsystem, and is started by the EXEC command

```
@XOFF (CR)
```

The program will print its name and version, then ask for the input file name:

```
INPUT FILE:
```

You should provide any standard TENEX file name specified in the usual way. You will then be asked to provide a name for an

```
OUTPUT FILE:
```

If you type an escape (altmode) as the first character, then the file name will be defaulted to

```
file.XGO
```

where 'file' was the name specified in the input file specification. Otherwise, you can type a TENEX file name in the standard way.

You will always be asked to confirm the name that you specified. If you type any valid confirmation character other than comma, several assumptions will be made. These are:

1. The output file will be formatted for output to the XGP
2. Underlining will be done by overstriking
3. The character set for the A partition is NGR9.XH
4. The character set for the B partition is BOD9I.XH
5. The paper height is approximately 11 inches

If you confirm the output file specification with a comma, you will be asked a series of questions, starting with:

```
Do you want the file formatted for the XGP (Y or N)?
```

A yes (Y) answer implies that you want to generate output to be sent to the XGP; otherwise a standard text file is assumed. Confirm this by typing Y (for yes) or N (for no). You will then be asked a series of questions depending on the type of output file being generated.

4.1 FILES BEING FORMATTED FOR THE XGP

If you are generating an output file to be sent to the XGP with the XGP program, you will be asked the following series of questions:

```
UNDERLINE PARTITION (A, B, O, or N)?
```

Partitions, character sets, and the like are discussed below. Basically, A or B means "use the A (or B) partition character set for any output marked as underlined," O (overstrike) really means underline, and N (No) means ignore underlining.

```
SWITCHES:
```

The last XGP-related question you will be asked is which SWITCHES you wish to specify. Reply with a carriage return only. Your input file is then processed.

WARNING: you should *never* list an XGP-formatted file on the line printer! There are special control codes inserted into the output file which cause justification, tabbing, etc. to occur on the XGP, but which will cause the line printer to do strange things like printing unrecognizable characters, performing random spacing and page ejects, and so on.

4.2 GENERATING STANDARD TEXT FILES

You will be asked the following questions if you are generating a non-XGP-destined file.

```
UNDERLINE (L, C, S, O, or N)?
```

L (line) is normal, meaning that the underlining will be done line by line rather than character by character. If you are generating output for the line printer, you should use the 'L' option. The options C (character), B (backspace) and O (overstrike) are intended for terminals with special facilities. N (no) causes underlining requests in the input file to be disregarded. The default if you type a carriage return or any character other than the above is overstriking.

```
SWITCHES:
```

Your answer is a carriage return.

```
SIMULATE FORM FEED (Y or N)?
```

There should never be any need to answer this question with anything but N (no). Its effect is equivalent to that of the same question in RUNOFF.

```
PAUSE?
```

Answer Y (yes) or N (no) as you would when using RUNOFF. "Yes" means "Stop the program at the completion of the processing for each page of paper." This is generally used when the output is going to an on-line output device so that each separate page of paper can be adjusted. Responding with N or carriage return is normal.

At the beginning of processing, XOFF will read the initialization file <XGP>INITIALIZE.XOFF (this may make the result differ from the un-initialized RUNOFF output of the same file, but it does all the standard things like single space and tab settings). When XOFF has finished processing your input file, it will type DONE and exit to the EXEC (@). You can then run the output through the XGP (or the printer, if that's where it's destined).

4.3 SOURCE FILE FORMAT

The source file contains the textual material which will appear on the final copy, plus information to specify formatting. All command information consists of regular ASCII printing characters so that a listing of the source file may be examined if the final copy is not exactly as desired.

All material in the source file is taken to be source text except those lines beginning with a period. A line beginning with a period is assumed to be a command, and must match one of those listed in the XOFF manual. The commands provide the formatting information, and control various optional modes of operation.

Usually the text is filled and justified as it is processed. That is, the program *fills* a line by adding successive words from the source text until one more would cause the right margin to be exceeded. The line is then *justified* by making the word spacings larger until the last word in the line exactly meets the right margin.

The user may occasionally wish to reproduce the source text exactly, which is done by disabling filling and justification. The program may be set to fill but not justify, in which case the output will be normal except that lines will not be justified to the right margin. The program may also be set to justify but not fill, although this would probably produce peculiar results and is not recommended.

When the fill mode is on, spaces and carriage returns occurring in the source text are treated only as word separators. A sequence of more than one separator is treated as a single separator.

Some of the commands cause a BREAK in the output. A break means that the current line is output without justification, and the next word goes at the beginning of the next line. This occurs, for example, at the end of paragraphs.

The program will advance to new pages as necessary, placing the title (if given) and the page number at the top of each page. The user may explicitly call for a page advance where desired, and may inhibit the occurrence of a page advance within specified material.

4.4 SPECIAL CHARACTERS

The character ampersand (&, shift-6) is used to specify underscoring. The ampersand will cause the character following it to be underscored, e.g. &f&co&co becomes foo.

Underlining of a string of characters can also be specified. An appearance of ampersand

preceded by up-arrow (↑&) will cause underlining of all following characters except space. An appearance of ampersand preceded by backslash (\&) will end this mode.

It is occasionally necessary to include spaces in the text which should not be treated as word separators. For this purpose, XOFF treats numbersign (#) as a quoted space; i.e., it will print as exactly one space in the output, will never be expanded nor changed to a carriage return.

To allow the appearance of the special characters (ampersand, number-sign, up-arrow, or back-slash) in the output, the character left-arrow (←, shift-O) is used as a quote character. The character immediately following a left-arrow will be transmitted to the output with no formatting effect. The left-arrow itself is just another case requiring the usual quote characteristic. The following five cases occur:

←&, ←↑, ←\, .←←, and

4.5 CHARACTER SETS, PARTITIONS, AND THEIR USE

In general, the only way to highlight things in bodies of text is to use capitalization, underlining, or some combination of the two. Using XOFF to generate output for the line printer, you still only have those facilities available to you, and in the same way as RUNOFF. However, when generating output for the XGP, you have much more flexibility in terms of the highlighting that can be performed, i.e., you have at your disposal a number of different fonts or character sets. Normally, you will need to utilize only two of them: one for your normal text and one for the same reason that you previously used underlining. Now, instead of underlining, you could use a special character font such as boldface or italics. However, you can utilize both underlining and a special font for purposes of highlighting.

The PDP-11 which runs the XGP has at most two character sets resident in its memory at any one time. Each character set resides in what is known as a partition, named the 'A' and 'B' partitions. Unless an XOFF command indicates which partition to use, the system assumes that the A partition is used for normal text, and the B partition is *not* used at all. The commands which will allow you to select which partition will be used to perform output are:

```
.USE A
.USE B
```

If you USE A or B, either the A or B partition will be selected.

Now, what if you wanted to use a total of three different character sets when generating output? A typical example of the usefulness of this is that a title and subtitles might look nice in a boldface type while highlighted words or phrases within the body of the text were italics. The commands used are:

```
.REQUIRE A KSET <filename>
.REQUIRE B KSET <filename>
```

where <filename> is the name of a character set known to the PDP-10. It can be the name of a file in your directory, or a file in <FONTS>. The name can be something like just "NGR9". The names of font files, by custom, are <style><height>.XH, where the height is given in printer's points (72 per inch). Thus, NGR9.XH is a font in "News Gothic Roman" style and is 9 points high. The available fonts are described in <FONTS>FONTS.DOC.

XOFF first looks in your directory for a file named NGR9.XH. If it can't find that file, then it tries <FONTS>NGR9.XH. This allows you to have private and specialized character sets, though it is not an easy job to create one. All of the sets you are likely to use can be found in directory <FONTS>.

4.6 DEALING WITH THE MARGINS

The length of a physical page generated by the XGP is based on the following parameters (settable by commands described in the manual):

```
Top margin (TM)
Bottom margin (BM)
Vertical space (VS)
Character height (CH)
The number of scan lines per page (LIN)
```

Whenever the PAPER HEIGHT (PH) command is issued, it looks at the current settings of TM, BM, CH, and VS, and computes the number of lines which will fit on a page. Since it is rare that an exactly integral number of lines will fit, the bottom margin is increased by the fraction of line size left over.

The use of the PAPER SIZE command should be avoided when formatting text for the XGP. This command should be used only for line-printer files. If you want both kinds of files, then use the following sequence of commands:

```
.IFX          (standing for "if XGP formatting")
.PAPER HEIGHT (and any other XGP commands you might want)
.ELSE
.PAPER SIZE   (and anything else depending on the LPT)
.FI
```

4.7 COMMANDS

The following commands are the most common; more can be found in the full XOFF manual. They will be recognized if they are at the beginning of a line started with a period. Any line in the source file beginning with a period is assumed to be a command. If it is not, an error diagnostic will be typed and the line will be ignored. Some commands take one or more decimal numbers or other arguments following. These are separated from the command by a space.

There may be more than one command on a line, separated by semicolons (with the exception of TITLE, SUBTITLE, CENTER, and INDEX which, when they appear on a line must be the last command on the line). For example:

```
.SKIP 3; CENTER
```

Commands which control the XGP will be ignored if an XGP format output file is not being produced; however, all such commands which break will do so regardless of the mode (XGP or normal).

Abbreviations or synonyms are listed below each command name. Because abbreviations are available, some illegal commands may be processed as other commands, e.g., "INEDNT" will be recognized as "I EDNT", i.e., it will index the word "ednt". The convenience gained by abbreviations was considered more valuable than the accidental recognition of an invalid command, since such errors show up very quickly.

.BREAK [n]
.BK [n]
.B [n]

Causes a break, i.e., the current line will be output with no justification, and the next word of the source text will be placed at the beginning of the next line.

.SKIP [n]

Causes a break after which n (line spacing) lines are left blank. (With the exception of TITLE, SUBTITLE, CENTER, and INDEX. These commands, when they appear on a line, must be the last command on the line.) If the skip would leave room for less than two printed lines on the page (i.e., if there are less than n+2 (line spacing) lines left), the output is advanced to the top of the next page. If n is omitted, 1 is assumed. If a SKIP command occurs at the top of an empty page, it is ignored. To force blank space at the top of a page, use FIGURE.

.BLANK [n]

Like SKIP but independent of line spacing; leaves n blank lines in output.

.FIGURE [n]
.FIG [n]

Like BLANK except that if less than n lines remain on the current page, the page will be advanced, and n blank lines will be left at the top of the new page. Principally used where it is desired to leave room for a figure to be drawn in manually. If n is omitted, a value of 1 is assumed.

.INDENT [n]
.IND [n]

Causes a break and sets the next line to begin n spaces to the right of the left

margin. The value of n may be negative to cause the line to begin to the left of the left margin (useful for numbered paragraphs).

If n is omitted, the indentation set by the last PARAGRAPH command (with argument) or the last PARIND command is used.

```
.PARAGRAPH [n]
.PARA [n]
.P [n]
```

The number is optional and, if present, sets the number of spaces used for paragraphs. The initial setting is 5. The command causes a break and leave (m+1)/2 blank lines, where m is the regular line spacing. The next line will be indented as indicated above.

```
.PARIND [n]
```

This command sets the indentation for subsequent PARAGRAPH commands. If n is omitted, a value of 5 is used.

```
.PAGE
.PG
```

Causes a break and an advance to a new page. Does nothing if the current page is empty. Titling and numbering as for automatic page advance.

```
.TEST PAGE [n]
.TP [n]
.NEED [n]
```

Causes a break followed by a conditional page advance. If there are n or more lines remaining on the current page, no advance is made and no lines are skipped. Otherwise, the page is advanced as for PAGE. This command should be used to ensure that the following n lines are all output on the same page.

```
.NUMBER [n]
.NUMBER UC [n]
.NUMBER LC [n]
```

Turns on page numbering (normal) and, if n is supplied, sets the current page number to n. Numbering may be done in decimal, upper case Roman numerals (UC) or lower case Roman numerals (LC). Page numbers are restricted to the range 0-999. Note that this command sets the current page number to n; thus the next page will be n+1.

```
.NONUMBER
```

Turns off page numbering. Pages will continue to be counted, so the normal page number will appear if numbering is re-enabled.

4.7.1 Mode Setting Commands

```
.JUSTIFY
.JUST
.J
```

Causes a break and sets subsequent output lines to be justified. Text processing always begins in JUSTIFY (and FILL) modes.

```
.NOJUSTIFY
.NOJUST
.NJ
```

Causes a break and prevents justification of subsequent output lines.

```
.FILL
```

Causes a break and specifies that subsequent output lines be filled. Sets the justification mode to be that specified by the last appearance of JUSTIFY or NOJUSTIFY. Text processing always begins in FILL (and JUSTIFY) modes.

.NOFILL
.NF

Causes a break and prevents filling of subsequent output lines. Also turns off justification. Note:

1. The nofill-nojustify mode need be used only where there are several lines of material to be copied exactly. A single line example will not require using these commands if there are breaks before and after.

2. Normally FILL and NOFILL are used to turn both filling and justification on and off. It is usually desirable to do both. However, a subsequent appearance of a justification command will override the fill command.

3. Because of the action of FILL, a single occurrence of NOJUSTIFY will cause the remainder of the file to be unjustified, with filling as specified. In order to justify but not fill (not recommended), a JUSTIFY command must follow every NOFILL command.

4. When a file is generated in XGP mode, no justifying is done by XOFF. Instead, XGP control commands are inserted in the file to instruct the XGP system to justify or not to justify. This is necessary since XOFF cannot justify properly for variable-width fonts.

5. Filling may be done by character count or by actual character width; see the KFILL and REQUIRE KSET commands in the XOFF manual.

.KFILL
.NOKFILL

In XGP mode these commands select or deselect character-set fill mode. In this mode, filling is done by using the actual width of the characters. Once this mode is set, it remains in effect for all subsequent FILL and NOFILL commands; consequently, the typical use of this command is once, at the head of a file. These commands are ignored in non-XGP mode. An implicit KFILL command is issued by any REQUIRE KSET command. These commands break.

```
.TURN ON c [ccc...]
.TURN OFF c [ccc...]
```

These commands activate or deactivate the listed control character(s). When a control character is turned off, it can no longer be used for its reserved function, and is treated as text. A control character may be reactivated with a TURN ON command. The control characters which are initially activated are:

↑ & ← # ,

The characters which may be additionally activated are: ?" ?# ?[(SOS inputs for certain special characters.)

?" ?# ?[@ $? ; :

```
.CONTROL
.NOCONTROL
```

The CONTROL command turns on the "standard" set of control characters, except sentence terminators, and NOCONTROL turns off all control characters except sentence terminators. The CONTROL and NOCONTROL commands are (respectively) equivalent to:

```
TURN ON ↑ ←  & #
TURN OFF ↑ ←  & # @ ?" ?# ?[ $
```

Note that NOCONTROL turns off more characters than CONTROL turns on. This is because the extra characters did not exist in earlier versions of XOFF. Note also that the sentence terminator characters are not affected by these commands.

4.7.2 Parameter Settings

.LEFT MARGIN [n]
.LM [n]

Causes a break after which the left margin is set to n. n must be less than the right margin, but not less than 0. The initial setting is 0. The amount of any indent plus the left margin must not be less than 0. If n is omitted, a value of 0 is assumed.

.RIGHT MARGIN [n]
.RM [n]

Causes a break after which the right margin is set to n, which must be greater than the left margin. If n is omitted, the current value of FIXED RIGHT MARGIN is used. The initial setting is 80.

The number of characters on a line will be equal to or less than the right margin minus the left margin minus any indenting which may be specified. Even if filling has been disabled, lines will not be extended past the right margin.

Within a footnote block the right margin is always set to the margin set by the last FIXED RIGHT MARGIN command. This allows the footnote to fill the entire line regardless of the width of the last line on the page. If a different size margin is desired for the footnote, appropriate commands within the footnote body may be used to change it.

.FIXED RIGHT MARGIN [n]
.FRM [n]

This command sets the "fixed" right margin. This is the margin to which page numbers are flushed and footnotes are justified. A RIGHT MARGIN command without an argument also sets the right margin to this value. This command breaks, and sets both the "fixed" right margin and the current right margin.

.ALTER LEFT MARGIN [n]
.ALM [n]
.ALTER RIGHT MARGIN [n]
.ARM [n]

Adds the value of n to the current value of the left or right margin. The value of n may be negative. These commands break.

Note that the margins may not be adjusted such that they cross (left margin greater than right margin) or exceed the paper size (less than zero or greater than the maximum right margin). Sometimes reversing the order of adjusting the left and right margins will prevent an illegal condition from occurring.

.SPACING [n]

Causes a break after which the line spacing will be set to n. The value of n must be within the range 1 to 5. Single spacing is 1, double spacing is 2, etc. (Initial setting is .SPACING 1).

.PAPER SIZE [n] [,m]
.PS [n] [,m]

(Note: this command is incompatible with XGP output; the PAPER HEIGHT command should be used instead.)

Sets the number of lines per page to n, which must be greater than 10. The initial setting is 58. The value of n includes the top margin of 5 lines. The page number and title appear on the second line. The second argument, m, is optional, but should be included if the default value is not used. If present, it sets the fixed right margin in columns. It must be greater than the left margin, and it is set into the right margin as if a FIXED RIGHT MARGIN m command had also been issued. This command is usually used only at the beginning of a file, but may be used throughout if needed.

.PAPER HEIGHT [n]
.PH [n]

(This should be used only for XGP output.) Sets the physical length of the paper to be n/100 inches. Default is 1100.

.TAB STOPS [n] ... [n]

Clears all previous tab stops and sets new tab stops as specified. The several n (max 32) must be greater than zero and in increasing order. They are the positions of tab stops independent of the setting of the left margin, although any which are less than the left margin will not be seen. Initial tab stops are 8, 16, ...

If a tab appears at a point where no further tab stops have been set on a line, the tab will be treated as though it had been a space.

.REQUIRE A KSET <filename>
.REQAKSET <filename>
.REQUIRE B KSET <filename>
.REQBKSET <filename>

Reads in the designated character set file and builds an internal table of character widths. When filling in KFILL mode, XOFF will fill lines by actual character width, rather than by character count. More than two character sets may be selected.

This command causes a KFILL command to be executed, and therefore causes a break.

4.7.3 Miscellaneous Commands

.TITLE [tttt ... tttt]

This command takes the remaining text on the line as the title. This text will appear at the top of all subsequent pages, at position 0, on the second line with the page number. The title is initially blank.

Note that the page number, which usually appears in the center of the page, is always forced to be beyond the title. Thus the page number may be forced to the right of the page by inserting blanks to the right of the title text. The word "PAGE" may be affixed by making it the last word of the title, appropriately separated from the rest of the title for readability.

If a TITLE command appears within the scope of a FLUSH RIGHT command, the page number will be forced flush to the right margin if blanks follow the title. If a TITLE command is not within the scope of a FLUSH RIGHT command, all blanks within the TITLE are converted to nonexpanding blanks.

To prevent the text of the title from being stretched when the page number is being forced right, the spaces within the TITLE should be nonexpandable blanks (, the quoted space).

The title is always printed using the character set and partition in effect when the TITLE command is issued. After printing the title, the character set and partition in effect at the end of the last page are restored.

```
.SUBTITLE [tttt ...  tttt]
```

This command takes the remaining text on the line as the subtitle. This text will appear on the line immediately following the title and page number. The subtitle is initially blank. The subtitle is not indented, but may contain leading spaces to achieve the same effect, if desired.

The subtitle is always printed with the same character set and partition as the title.

```
.CENTER
.CEN [n] [,m]
.C [n] [,m]
```

This command causes a break after which it centers the next line following in the source file. The centering is over the column n/2, independent of the setting of the left and right margins. If n is missing, n is assumed to be the fixed right margin value, initially 80.

The optional argument, m, specifies that the next m lines (including blank lines) are to be centered. If m is omitted, it is assumed that one line (i.e., the next line) is to be centered.

Note that the argument m is a specific number of lines. These lines must not contain commands; any line which begins with a command character (usually a period) will terminate centering. No error message is issued. This feature is convenient when centering an undetermined amount of text, or where counting would be tedious. A "CENTER ,999" command can be issued, and the centered text terminated with a (possibly empty) command line.

.FOOTNOTE [n] or [-n]
.FOOT [n] or [-n]

Allocates lines at the bottom of the current page for a footnote. If the value of n is positive, then n*(line spacing) lines are allocated; if n is negative, then abs(n) lines are allocated (hence, negative numbers indicate the space allocated is to be independent of the spacing in effect at the time the command is issued). If insufficient room remains on the current page, space will be allocated at the bottom of the following page. The text for the footnote should begin on the line following the command, and it may contain any appropriate commands (e.g., CENTER, SKIP) necessary to format the footnote. The footnote is terminated by a line beginning with an exclamation point (the remainder of which is ignored). The lines delimited by this line and the FOOTNOTE command are put into a buffer to be processed when the output moves to within the stated distance of the bottom of the page. If a page has multiple footnotes, the allocated space is the sum of the allocations for all footnotes assigned to the page. The user must include his choice of footnote-designating symbols within the text.

An implicit block (block name *FOOT) surrounds each footnote. Thus parameters such as margins, indentation, spacing, character set in use, etc. are restored at the end of the footnote and may be altered within the body of the footnote without influencing any other part of the document.

Before processing the body of the footnote, an implicit sequence of commands is executed:

```
.BEGIN *FOOT
.FILL; NORETAIN; LEFT MARGIN 0; RIGHT MARGIN
.CONTROL; COMMAND CHARACTER; UNDERLINE OFF
.PARIND 5
```

This prevents undesirable side effects, e.g., if the last line of the page was in FLUSH RIGHT mode, and if this mode were not turned off within the footnote block, the footnote text would be forced flush right.

The actual space taken by the footnote may be more or less than that specified by n. The n merely allocates space and should be the user's best guess. If it is considerably off, the footnote lines may overflow the page, or extra space may be left at the bottom. The user may wish to adjust n after examining a first draft printout.

.FSEP [n]

This command allows the user to specify a sequence of commands to be processed after the "implicit" set (which always precedes footnote processing) and before the actual processing of the footnotes. The sequence of commands follows and, like the body of a footnote, is terminated by an exclamation point (!) in the first character position of a line.

This sequence of commands is processed once before the footnotes are placed on the page, and usually is used to provide a "separator" sequence, such as a blank line followed by a dashed line, etc. The argument, n, is the number of lines which the text of the FSEP body requires. Footnote separators are *not* affected by block structure; they remain in effect until explicitly changed.

.INDEX [tttt ... tttt]

This command takes the remaining text on the line as a key word or words and adds it, along with the current page number, to the internal index buffer. The command does not cause a break. It should appear immediately before the item to be indexed. A key word or words may be indexed more than once.

.PRINT INDEX

Causes a break after which it prints the entire contents of the index buffer. Entries are printed in alphabetical order, and are set against the left margin. The user may exercise control over the ordering; it need not be the ASCII collating sequence. See the COLLATE IN command. Regular line spacing is used, except that a blank line is left between entries of different first letters.

The actual determination of "difference" is under the control of the user; see the COLLATE OUT command. The number of the first page on which each entry appeared is put on the same line as the entry, beginning at the middle of the line (midway between the left and right margins). Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

.COMMAND CHARACTER [c]
.CC [c]

This command sets the character which is used in the beginning of a line of commands. Normally it is a period (.), but it may be changed to any character except a semicolon by use of this command.

.BEGIN [ttttt]
.END [ttttt]

The idea of block structure is that any changes to margins, spacing, etc. are local to the block and will be restored to their original values upon exit from the block. The XOFF block makes all variables except the TITLE, SUBTITLE and page number local to the block. In addition, any XGP commands issued in the block will be undone when the block is exited. Blocks may be nested, and an implicit block surrounds footnotes.

As an aid to the user in assuring that blocks are properly nested, any 5 characters (not including a semicolon or asterisk) may be included after a BEGIN. This is the block name for the block defined by that BEGIN. The END which matches the BEGIN must have the same block name following it as well, or it is flagged as an illegal command.

Two special blocks are generated by XOFF: "*OUT*" is the outermost block which is global to the entire text file, and "*FOOT" is the block which surrounds footnotes.

5. RUNFIL PROGRAM

The RUNFIL program provides a means for using a file instead of a terminal to supply an input command stream. A file is prepared containing the character stream which would be typed on the terminal to perform some desired set of operations. The stream is initially received by the EXEC but may start any subsystem, and primary input will continue to come from the file. All output is directed to the controlling terminal in the usual manner, and the resultant typescript is indistinguishable from that produced when input is from the keyboard. Optionally, output may be directed to a file other than the terminal. However, such a file may contain "errors" in echoing, that is, the source file "echoes" present in the output file may be out of phase with the output.

```
@RUNFIL (CR)

COMMANDS FROM FILE:  <filename>
```

The user supplies the name of the file from which commands are to be taken. If the name is confirmed with carriage return, output will be directed to the terminal and processing will begin. If the name is confirmed by a comma, the program will ask:

```
OUTPUT TO FILE:      <filename>
```

and the name of the file to receive primary output should be entered.

The program begins by starting an EXEC in an inferior fork, with primary input and output set as determined by the dialogue above. The command file must contain EXEC commands to start any desired subsystem. The last command should leave control at the command level of the EXEC. When the end of the file is reached in the command file and any processing is completed, the inferior fork(s) will be killed and control returned to the top-level EXEC.

5.1 FORMAT OF COMMAND FILE

As stated, the command file contains exactly what would be typed on a controlling terminal to perform some desired set of operations. To facilitate certain operations, however, the following actions cause special interpretations by RUNFIL.

Control-↑ acts as a warning character and interprets the immediately following character(s) as follows:

1) A-Z, [,\,],↑,←,@

Converts the single character in this group to its control equivalent, e.g., ↑C (i.e., control -↑ followed by C) becomes ↑C, etc.

Note: Control-C in an input stream causes an immediate termination of processing as usual. However, for some subsystems, ↑C is the only way to return to the EXEC, and the user would normally not type ↑C for this purpose until processing had been completed. In a command file, ↑↑B serves this function, ie. RUNFIL waits until the inferior program is ready for more input (an indication that processing is completed), and then sends a ↑C.

2) DECIMAL NUMBER

The number will be taken as a length of time (in milliseconds) to wait before taking further input from the command file.

The number must be terminated by a non-numeric character (e.g., space), which will have no other effect in the input stream. This "pause" facility is furnished to allow orderly input from the terminal at desired points in the file stream.

5

Note that once processing has begun, any characters typed on the controlling terminal will be intermixed with the file stream, usually producing undesired results. In particular, a ↑C typed on the controlling terminal be be handled by the lower-level EXEC, and so is not a way to abort command file processing. The character ↑B is enabled in RUNFIL for this purpose. (↑B is the only character which has a different effect when input from the terminal.)

In most cases ↑B from the terminal will immediately terminate command-file processing and return to the top-level EXEC. However, ↑B typed during a requested "pause" in command-file input (↑↑<decimal number> above) will break the pause, ie., RUNFIL will "wake up" and immediately resume command file input. Thus processing can be caused to continue after manual terminal input without waiting out the remaining specified delay. An actual ↑B (as opposed to ↑↑ followed by B in the file stream) is treated as ordinary input and is passed to the subsystem running under RUNFIL.

6. TENEX CALENDAR PROGRAM

6.1 INTRODUCTION

The CALENDAR subsystem is a program which is especially for people who like to keep lists of things to do daily. It is also useful for people who need reminders of things like meetings, appointments, and so forth.

The CALENDAR subsystem gets its commands from the terminal. The commands are very simple and are designed to handle those operations most frequently performed on the calendar data. They do not constitute a general set of primitives which one might consider the basis for programs to operate on calendar data. CALENDAR is not intended to be a programming language.

6.2 ENTERING INFORMATION IN THE DATA BASE

There are two primary commands for entering appointments, deadlines, and things to do in the data base; the Enter command and the Appointment command.

When the Enter command is invoked (by inputting E on the terminal), CALENDAR asks for the date which is terminated by carriage return. Dates are read by the TENEX time and date monitor calls which permit relatively free format to be used. If the date input is null (only a carriage return is typed), the current date is assumed. Whenever CALENDAR asks for a date, the preceding comments apply. Next CALENDAR types a task number (maximum of 256 tasks per day) to be assigned to this task or thing-to-do. This number is *useful only for addressing the task at a later time, you need not remember* it because it is always output when CALENDAR types a reminder or a task. Next CALENDAR waits for the user to input an arbitrary length description of the task. Editing may be done on this description with control A or control Q consistent with standard TENEX editing. A carriage return may be put into the description for multi-line tasks. Tasks are terminated by ↑Z or ESC (altmode key).

A normal Appointment is input by typing A on the terminal. CALENDAR will now ask for the date of the appointment, the number of days between reminders you want to see before the appointment, the total number of reminders you want to see, and finally some arbitrary length text about the appointment or deadline. It should be noted that appointments are considered by CALENDAR to be simply tasks which have some reminder criteria. All operations which can be done on tasks can be done to appointments.

Another type of appointment may be input via the Appointment command. This is a so-called "forward" appointment for which you would like to see a reminder every specified number of days (up to a particular count) starting from a given date and going forward in time. This is specified by terminating the date field of an appointment command by a left arrow, carriage return (or ↑Z or escape) sequence.

Forward reminders are distinguished in task typeouts (see List command) by an "" preface. When the number of times a forward reminder has been given equals the requested count, the preface is changed to an "=" and the reminder is listed every day until it is explicitly marked as finished, cancelled, or deleted in the normal fashion.

The reminder typeouts are very diligent about reminding you when you asked. For example, if you asked for 4 reminders 4 days apart and then didn't use CALENDAR until 10 days before the deadline, you will get reminded this time as well as the very next time you use CALENDAR so that it can "catch up". This is accomplished by keeping a count of the actual number of reminders given. A reminder is unconditionally forced on the work day before an appointment or deadline. CALENDAR knows about weekends but not holidays...

If the input for the total number of reminders you want to see is null (just carriage return typed in), CALENDAR will put in a count sufficient for enough reminders to cover the entire period from the current day to the deadline at the frequency you specified.

6.3 UPDATING THE CALENDAR DATA

The Enter and Appointment commands will not have permanent effects on the data base until the Update command is given. This is true of all commands that modify the data base. The output by CALENDAR of a reminder modifies the data base and an Update should be done to have a permanent effect. Update requires a confirming carriage return before it is executed.

6.4 RELATIVE DATES

When CALENDAR asks for a date, you can type in a date relative to the current date by typing a small number (1-9). If you want this to be a relative number of workdays (Saturday and Sunday will be skipped over), preface the small number with colon (:). For convenience, : alone is equivalent to :1.

The character "." is used to mean the most recently typed-in explicit date (i.e. 5/15/73 and

null are explicit date type-ins, :3 is an implicit date meaning plus three work days. Explicit date typeins re-establish the meaning of "." Implicit date typeins have no effect on the meaning of "."). It is possible to use signed, small numbers as arguments to the ":" relative date operator. Signed small numbers also work as arguments to "." the sequence :.-2 is interpreted to mean two work days before the last explicit date typed in. Most combinations of the date operators which make sense are accepted and do what you might expect.

6.5 LISTING THE CALENDAR DATA

Selected portions of the data base are listed by the List, Total list, and Individual list commands. The normal command used is List. List asks for a date, and normally a null input should be typed which means to use the current date and to invoke the standard reminder option. This will result in the current date and time being output followed by a list of reminders (if any) followed by a list of tasks (if any).

Each reminder or task is prefaced by the date of the task and its task number; then the text associated with the task is output. The preface date field is omitted if the task is for the current date. Reminders and tasks are always output in chronological then increasing task number order. Tasks should be marked as either Finished, Cancelled, Rescheduled, or Deleted to get them to disappear from the List printout. If you don't do one of these, the List printout of the task is repeated in the task list ad nauseum. The List output can be directed to a file by using the Output command. Output is initially set to the terminal. The standard reminder option causes the reminder counts to be updated. Recall this is invoked by inputting a null date. If List is given a date input, the reminder counts are not updated; instead, the CALENDAR program outputs reminders and task lists as though the current date were the date that was input. Note that if this is a future date, all unfinished tasks through that date are output. Appointments and deadlines are both treated as tasks (except that they are prefaced by an "!" for normal appointments or "" or "≥" for "forward" reminders), and this becomes obvious as they jump from the reminder to the task list as the List date is the same or greater than the appointment date.

There are times when you want to see only the tasks for a given date with no reminders of old, unfinished tasks. This is done by using the Individual list command which is otherwise exactly like List. There are also times when you want to see everything including Finished tasks and Cancelled tasks. This is accomplished by using the Total list command. With this command a single-character indicator prefaces finished tasks (F) and cancelled tasks (C). Appointments are prefaced by three numbers of the form (l,m,n) where l is the number of days between reminders, m is the number of reminders, and n is the current count of reminders issued.

6.6 FORMAT CONTROL

CALENDAR normally outputs multi-line tasks in a "balanced format" which tests to see if words will fit on lines and inserts a carriage return,line feed if the word would over run the line. In the normal mode all explicit carriage returns are translated into spaces. This mode may be turned off by use of the "Balancing Format Switch ON (Y or N?: " command - answer that question N to turn the mode off. Within each task the user has some degree of multi-line format control. The first ↑V (control V) encountered in a task turns balance formatting off. Each successive ↑V complements the state of balancing format from off to on, on to off... Remember that ↑V is the "quote next character" control so to input one, ↑V must be itself input twice. Also, when balanced formatting is off, ↑V is ignored.

6.7 THE PRINT COMMAND

The "Print" command is available for printing sections of the calendar data -- such as for a month at a time. It prints a week at a time per page (which may over run a page for a busy week). On hard copy controlling terminals, it will await the input of any character (such as a space) between pages to allow the user to tear off output. Print expects the user to input a beginning and ending date for the period. If the first date is null (default for the current date), a standard "List" command for that first date will be executed including reminders and any incomplete tasks before the current date. If an explicit date is typed for the first date, a listing will be made for the first date only for that date with no reminders. Successive dates are typed with only the data for that date with no reminders.

6.8 MODIFYING THE CALENDAR DATA

Tasks or appointments are marked as finished with the Finished command which takes a task address of the following form: task number, date. They are marked as cancelled by the Cancelled command which takes a task address of the same form. The Delete command takes a task address and marks the space to be reclaimed during the update process. The Gain command is used to reclaim space in a similar fashion for all tasks with a date as old or older than the date supplied. Deleted and gained tasks literally disappear from the data base. Gain will optionally gain space only for completed or finished entries.

There is a provision for rescheduling tasks with the Reschedule command which marks the original entry as cancelled (it's still there) and makes a copy of the entry with the newly specified date.

An individual task may be itself modified with the "Modify" command. This expects a task number and date. It then invokes Teco which is used to edit the task. When you are finished, type ";H$(alt-mode)" to Teco to get back to CALENDAR.

6.9 RETURNING TO THE EXEC

The Quit command (if confirmed by a carriage return) will get you back to the EXEC. Quit closes any opened output file and switches output back to the terminal for subsequent CONTINUE's. CONTINUE will get you back into CALENDAR. Do not use re-enter; for one thing there is no re-enter address and TENEX won't let you.

6.10 ENCRYPTING THE DATA

The Key command enables a change in the optional Key for the encrypting of the data base in a way that is believed to be relatively crack-proof. The encrypting algorithm uses this key to form the basis or seed of a double-word pseudo-random number generator sequence. Successive high order parts from the generator are exclusive-Ored with the data thus encrypting the data. A checksum is included with the data to be encrypted. This is used on input to determine if the user typed in the correct Key.

If you are down on keys, passwords, and privacy..., then don't specify a key. When you first use CALENDAR it asks you whether or not you want the data encrypted. Of course you can always encrypt or change the key with the Key command. Don't forget the key! It is very likely the data will be impossible to decrypt without it! If you want to go from encrypted to non-encrypted data, type a null key (just CR) to the Key command.

With encrypted data, CALENDAR will ask you the key whenever you use the CALENDAR subsystem. If you mistype the key, you are returned to the TENEX EXEC.

6.11 MANAGING OTHER CALENDARS

The Yank file command enables the user to specify another file for CALENDAR to work with. This can be a file in the connected directory or any other directory to which the user has access. Simply input a standard TENEX file name after the Yank file command is given. This command is most useful for a secretary to manage her supervisor's calendar. The default fields are set to "<connected-directory>CALENDAR.DATA;1".

6.12 MISCELLANEOUS COMMANDS

There is an Xor command which allows the undoing of deletes,finishes, cancels, etc. This is accomplished by specifying a task and date then typing D for delete,F for finished,C for cancelled in any combination to accomplish an exclusive OR of that operation with the present state of that operation for the specified task; thus, you can also mark a task, for example, as finished with this command.

6.13 INTERRUPTING CALENDAR

The rubout key is enabled in CALENDAR to abort operations and output whenever it is safe to do so. Control C followed by CONTINUE is safe as with all subsystems. Reenter would not be safe and is not allowed.

6.14 THE DATA BASE

The data base is stored in the users file CALENDAR.DATA;1. This is an ordinary TENEX file, and if it is deleted, it will go away in the ordinary way. (This is by contrast with MESSAGE.TXT;1). The file is read into the address space of CALENDAR when it starts up. This makes data base references very fast but limits the maximum usable size of the file to 246K data words. This is effectively infinite, but if it gets nearly full, the user will be so warned and expected to make use of the Delete and/or Gain commands. Of course it can be Total listed prior to doing this. Users should be aware that many files of this size tax disc space enormously. Since CALENDAR.DATA;1 is an ordinary file, accounting for its space is done in the standard way.

The update operation was written to immunize the data base from crashes. Namely the file CALENDAR.NEW;1 is used to write the updated version then this is renamed to CALENDAR.DATA;1. Of course like any file, it may be wise to back it up and/or list it periodically.

6.15 AN EXAMPLE OF USING CALENDAR

```
Calendar 6-11-73      THU 12/27/73 15:26:11
Do you want to encrypt your data (Y or N)?: Y
Key is: COMPACT
If you forget this key, your data is likely to be irretrievably  lost!  It
is printed here again, enclosed in square brackets.
[COMPACT]
 Appointment or deadline for date: January 7, 1974
Number of days between reminders: 1
Number of reminders: 3
Task(1): 1400 PLANNING MEETING
 Enter task for date: 12/31/73
Task(1) TURN IN TIME SHEET$
 Enter task for date: .
Task(2): FINISH INPUT FOR ARPA QPR$
 Enter task for date: .+1
Task(1): HAPPY NEW YEAR, OUT ALL DAY TODAY$
 List tasks date: 12/28/73
THURSDAY, DECEMBER 27, 1973 15:29:49-EST for FRIDAY, DECEMBER 28, 1973
 List tasks date: 12/31/73
THURSDAY, DECEMBER 27, 1973 15:29:59-EST for MONDAY, DECEMBER 31, 1973
     1 TURN IN TIME SHEET
     2 FINISH INPUT FOR ARPA QPR
 List tasks date: 1/1/74
THURSDAY, DECEMBER 27, 1973 15:30:12-EST for TUESDAY, JANUARY 1, 1974
12/31/73 1 TURN IN TIME SHEET
12/31/73 2 FINISH INPUT FOR ARPA QPR
         1 HAPPY NEW YEAR, OUT ALL DAY TODAY
 Individual Listing date: Jan 3, 1974
```

```
Want to see F and C entries (Y or N)?: N
Want to see reminders (Y or N)?: Y
THURSDAY, DECEMBER 27, 1973 15:30:42-EST for THURSDAY, JANUARY 3, 1974
 Individual Listing Date: Jan 4, 1974
Want to see F and C entries (Y or N)?: N
Want to see reminders (Y or N)?: Y
THURSDAY, DECEMBER 27, 1973 15:30:53-EST for FRIDAY, JANUARY 4, 1974
**Reminders**
! 1/07/74 1 1400 PLANNING MEETING
 Update [Confirm]
 Quit [Confirm]
```

NOW CALENDAR IS CALLED AT A LATER SESSION

```
@CALENDAR
 Calendar 6-11-73     THU 12/27/73 15:31:35
Key is:
 Cancel Task Number: 1 Date: 1/7/74
 Reschedule task number: 2 Date: 12/31/73
to new date: 1/2/74$
 Update [Confirm]
 List tasks date: 1/2/74
THURSDAY, DECEMBER 27, 1973 15:33:15-EST for WEDNESDAY, JANUARY 2, 1974
12/31/73 1 TURN IN TIME SHEET
 1/01/74 1 HAPPY NEW YEAR, OUT ALL DAY TODAY
         1 FINISH INPUT FOR ARPA QPR
 Quit [Confirm]
@
```

7. FILE TRANSFER PROTOCOL (FTP)

FTP (File Transfer Protocol) provides facilities for file transfer between hosts on the ARPANET. The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently.

7.1 FTP COMMAND INTERPRETER

Instructions to the FTP program are given via the FTP Command Interpreter. Characters typed on the user's terminal are read by the FTP Command Interpreter and decoded as commands to perform various actions by FTP.

Typing a "?" to the FTP Command Interpreter will yield a message to use the "HELP" command to type a summary of the FTP commands.

The FTP Command Interpreter provides command completion whenever a terminator is typed (full-duplex terminals only) and an exact match is achieved with some command or a unique initial substring is typed. Terminators are space, comma, alt-mode (ESC), and carriage return. Terminators are often not distinguished and are thus equivalent. Where necessary, comma is used to separate list items, space terminates a command or option and signals the desire to specify more options, carriage return ends a command unless more information is necessary. Altmode (ESC) is the same as space except that it will cause command completion in those modes where it is normally suppressed.

7.2 ESCAPING BACK TO EXEC MODE

At any time, typing a Control-C (↑C) will cause FTP to stop whatever it is doing and return to the EXEC mode.

7.3 DATA TRANSFER FUNCTIONS

Data and files are transferred only via the data connection. The transfer of data is governed by FTP data transfer commands received on the FTP connections. The data transfer functions include establishing the data connection to the specified host (using the specified byte size), transmitting and receiving data in the specified representation type and transfer mode, handling EOR and EOF conditions, and error recovery (where applicable).

7.4 MAKING A CONNECTION

There are two ways of making a connection. Typing "CONNECT host-name" or "CONNECT octal-number" will cause a connection attempt to be made. If successful, the connection will be said to be complete. If unsuccessful, the connection will be said to be incomplete with a reason given.

7.5 DISCONNECTING

There are three disconnect commands. "DISCONNECT" disconnects the user from the remote host without returning to the EXEC, "BYE" is the same as "DISCONNECT," and "QUIT" returns the user to EXEC without closing the connection. Thus, to close the connection and return to EXEC, the user should type the "DISCONNECT" command followed by the "QUIT" command.

In the event that the network connections are severed by a network failure, the user will receive a message that the connection has been severed and/or that the data transfer is incomplete.

7.6 COMMAND SUMMARY

ACCESS CONTROL COMMANDS

CONNECT host-name or octal-number

Performs ICP to connect to the indicated host. This must be the first command issued.

LOGIN username optional-password optional-account

User identification that is required by the server for access to its file system.

CWD anothername

CWD is used to change the working directory to anothername. Using it requires the "QUOTE" command at present. (Example: *QUOTE CWD anothername.)

ACCOUNT number or string

The argument field is a number or ASCII string identifying the user's account.

DISCONNECT

Disconnects user from remote host without returning to EXEC.

BYE

Same as DISCONNECT.

QUIT

Returns user to EXEC without closing connection.

SERVICE COMMANDS

GET REMOTE-FILE to LOCAL-FILE

The REMOTE-FILE is transferred from the host site to the user's site and is given the LOCAL-FILE name. Note: REMOTE-FILE and LOCAL-FILE use standard filename formats.

SEND LOCAL-FILE to REMOTE-FILE

The LOCAL-FILE is transferred from the user's site to the host site and is given the REMOTE-FILE name. Note: LOCAL-FILE and REMOTE-FILE use standard filename formats.

MULTIPLE GET/SEND

Only TENEX sites permit the use of "*" in the specification of filenames for GET and SEND. The standard "*" formats must be used.

APPEND LOCAL-FILE to REMOTE-FILE

The LOCAL-FILE is transferred from the user's site to the host site and appended to the REMOTE-FILE at the host site. Note: LOCAL-FILE and REMOTE-FILE use standard filename formats.

RENAME REMOTE-FILE to be NEW-REMOTE-FILE

The filename for REMOTE-FILE is Changed to NEW-REMOTE-FILE. Note: REMOTE-FILE and NEW-REMOTE-FILE use standard filename formats.

DELETE REMOTE-FILE

This command deletes the REMOTE-FILE. Before the file is actually deleted, the user is asked "Do you really want to delete? (Y or N)". Note: REMOTE-FILE uses standard filename formats.

DIRECTORY of REMOTE-FILE-GROUP

A list of the files in the REMOTE-FILE-GROUP will be typed out (e.g., <SMITH>*.MAC, if remote site is a TENEX site).

STATUS of remote-system

Status information about the remote-system will be typed out.

MAIL <file> to REMOTE-USER

Sends LOCAL-FILE to mailbox at remote site.

HELP

Types summary of FTP commands.

MISCELLANEOUS COMMANDS

VERBOSE

Types out all comments in long form.

BRIEF

Types out all comments in short form.

QUOTE arbitrary-FTP-line

Sends arbitrary-FTP-line to remote-site without interpretation.

STATISTICS

Turns on typeout of timing statistics.

NOSTATISTICS

Turns off typeout of timing statistics.

CONTROL CHARACTERS

↑G (Control-G)

Type BELL (↑G) to abort a file transfer and return to command level.

↑O (CONTROL-O)

Type ↑O to clear typeout buffer.

↑V (Control-V)

Use ↑V to quote characters in LOGIN.

7.7 EXAMPLE OF FTP USE

```
@FTP                                ;call in the subsystem
*BBN                                ;connect to host BBN
*LOG SMITH SECRET 12345             ;declare name, password, account
                                    ; the password will not be echoed.
*DIR *.MAC
(to local file) TTY: [confirm]      ;get a partial directory listing
*GET PROGRAM.MAC                    ;must end with carriage return
(to local file) <esc>PROGRAM.MAC    ;escape causes same name to be used
*DISCON                             ;break the network connections
*QUIT
@
```

8. TELNET USER GUIDE

User Telnet (hereafter called Telnet) provides facilities for communicating with host computers via the ARPA network utilizing the TELNET protocol. The purpose of the Telnet program is threefold. It converts various terminals connected to TENEX into a standard type of terminal called a network virtual terminal (NVT) by interposing programs in the character streams between the terminal keyboard and printer and the terminal port on the host computer. Secondly, it provides information about the network to assist a user in establishing connections. Thirdly, it multiplexes the terminal among several remote jobs.

8.1 TELNET COMMAND INTERPRETER

Instructions to the Telnet program are given via the Telnet Command Interpreter. When in command mode (see below), characters typed on the user's terminal are read by the Telnet command interpreter and decoded as commands to perform various actions by Telnet.

The Telnet command interpreter has two unique features. The command interpreter will refuse to hear anything it does not understand. With full-duplex terminals, this means that no echo will appear for characters which are not valid successors of the previous input. In any case, the character is ignored and a bell is typed out. The input stream that has already been typed is not forgotten, however. Therefore, it is only necessary to type the correct character and not the complete command. This feature may be turned off with the "no fancy.command.interpret" command.

The other unique feature of the Telnet command interpreter is the use of question mark to discover what the command interpreter expects next. Typing a "?" at any time in command mode will elicit a list of words the command interpreter is expecting. Thus, typing a "?" when nothing has been typed will yield a list of all possible top-level commands. Typing "co?" will yield a list of all commands starting with "co". Typing "connection.to ?" will yield a list of possible arguments to the "connection.to" command.

The command interpreter provides command completion whenever a terminator is typed (full-duplex terminals only) and an exact match is achieved with some command or a unique initial substring is typed. Command completion may be suppressed with the "concise" command. Terminators are space, comma, alt-mode, and carriage return. Terminators are often not distinguished and are thus equivalent. Where necessary, comma is used to separate list items, space terminates a command or option and signals the desire to specify more options, carriage return ends a command unless more information is necessary. Altmode is the same as space except that it will cause command completion in those modes where it is normally suppressed.

8.2 COMMAND/REMOTE MODE

As mentioned above, characters typed on the terminal keyboard may be used in two ways: either as commands to Telnet, or as input to the remote host. The choice is made on the basis of whether Telnet is in remote mode or command mode. In command mode, characters typed on the terminal keyboard are read by the Telnet command interpreter and decoded as commands to perform various actions. TELNET is initially in command mode and will revert to command mode whenever the Telnet escape character (see below) is typed.

The opposite of command mode is remote mode. In remote mode, characters typed on the keyboard (with certain exceptions) are not examined by Telnet at all, but are merely passed on to the remote host computer. Remote mode is normally entered after any command is executed when the current connection exists. The "local.mode" command may be used to defeat this. The effect of the "local.mode" command is cancelled by the "remote.mode" command or by the "connection.to" or "retrieve.connection" commands.

8.3 ESCAPING BACK TO COMMAND MODE

At any time, typing the Telnet escape character (initially control-Z (SUB)) will cause Telnet to stop whatever it is doing and return to command mode. Occasionally, a slight delay may be experienced due to the need to clean up whatever was happening at the time. Telnet announces the switch to command mode by the appearance of a sharp sign "#" at the left margin. Telnet also indicates the transition out of command mode by the appearance of another sharp sign followed by a new line.

WARNING: If you have control-Z anywhere in your programming, you should change your escape character for Telnet to other than control-Z to avoid mishaps.

8.4 MAKING A CONNECTION

There are two ways to make a connection. Typing "connection.to <host> [<qualifiers>]" or simply typing "<host> [<qualifiers>]" will cause a connection attempt to be made. If successful, the connection will be said to be complete and the terminal will be placed in remote mode. If unsuccessful, the connection will be said to be "incomplete because ---" with a reason given; also if the remote host is down, a line is typed telling why and for how long. By terminating the host name with a space, one or more qualifiers may be specified. Ordinarily socket 1 is assumed. Thus without a qualifier, the connection will be made to the "logger" on the remote system. By using an octal number as a qualifier, the connection

will be made to the socket so specified. A set of names is available for specifying the socket desired. This set consists of names for all the standard sockets.

The "wait" qualifier may be used to camp-on the connection. This qualifier causes Telnet to repeat the attempt to connect in the event of a failure until it finally succeeds. An initial failure causes a message to that effect to be printed. When the attempt finally succeeds, bells are typed out to wake the user up. The attempt to connect may be aborted by typing the Telnet escape character.

The "load.settings.from..." qualifier (possibly qualified with "no") may be used to cause (inhibit) the mode flags to be initialized from the mode file. When inhibited, the current modes are used.

The "name.for.connection" qualifier may be used to specify a name for this connection other than the one assigned by Telnet. A name for the connection may also be given later by the "name.for.current.connection" command.

8.5 DISCONNECTING

The "disconnect" command is used to close the current connection. This will not necessarily log you out from the remote host so you should perform the logout procedure for that host before disconnecting. The disconnect command takes an optional argument specifying the name of a particular connection to be disconnected. See multiple connections and connection names below.

In the event that the network connections are severed by a network failure, the message "IO error for connection <name>" is printed, the connection is disconnected, and Telnet reverts to command mode. This may happen even if the error occurs on a connection which is not current. If the remote host initiates a disconnect, a message to that effect is printed and the same action is taken.

If the remote host on the current connection stops responding when input is being sent, a line is typed, "Host not responding on connection xxx." (In this case the connection is *not* lost.) When the remote host resumes operating, the user is informed: "Service restored on connection xxx."

8.6 ECHO CONTROL

Telnet allows several options concerned with echoing. Echoes may be generated by the

terminal, by Telnet, or by the remote host. Telnet determines if the terminal is generating echoes when started by examining the mode word for the terminal. The "terminal.type.is" command may be used to change this.

If the terminal is echoing, then Telnet will do everything possible to cause the remote host to not generate echoes, and Telnet will not generate echoes itself. If the terminal is not generating echoes, then Telnet determines whether it should echo or not by information in the mode file (if any) or by the "echo remote"/"echo local" commands, or by information sent from the remote host.

Telnet keeps the remote host informed about how echoing is being done; if the remote host is suitably equipped, it will follow along. If not, then the user will have to give commands to the remote host to achieve the proper echoing. Telnet also will respond to commands from the remote host concerning who should be echoing. If Telnet believes the terminal is doing its own echoing, it will respond to any request from the remote host to not echo by an "I'll echo" command.

8.7 LINE BUFFERING AND END OF LINE CONVENTIONS

Telnet provides an optional line buffer for use with line-oriented operating systems. In this mode, characters typed in remote mode are stored in a local buffer up through an end of line. Prior to the end of line, the currently buffered line may be edited using control-A (SOH) or control-H (BS) to delete characters, control-X (CAN) to delete everything, and control-R (DC2) to retype the current contents. Telnet always converts the TENEX EOL into the NVT EOL. TENEX in turn converts a carriage return into the TENEX EOL. Thus typing a carriage return will cause the buffered line to be transmitted. Linefeed may also be used to terminate a line. In this case, the transmitted line will end with only linefeed, not the NVT EOL.

Telnet provides an optional linefeed echo for carriage return. If the remote host provides a linefeed also, then the echo generated by Telnet should be suppressed with the "echo no linefeed.for.carriage.return" command. In remote echo mode, Telnet generates no echos whatsoever. In this mode, all echos must be provided by the remote host.

8.8 STATUS COMMANDS

Several status commands are available for discovering facts about the network. None of these commands will affect the state of the current connection. The status commands include where.am.I, status.of, netstatus, and socket.map. These commands are summarized below.

8.9 SPECIAL CHARACTERS

Several commands are available to send characters which do not appear on the terminal. "Code" takes an octal (decimal if preceded by "D", hexadecimal if preceded by "H") argument and sends the character with that code. The word "code" may be omitted and just the argument typed. "Control" takes a character argument and sends the corresponding control character (the low order five bits of the character) is sent. The "!break!" command sends the NVT break character, which is mapped by some systems into the equivalent of the attention, quit or break key which appears on some terminals.

To facilitate operation with systems requiring frequent use of special characters or lower/upper case graphics which a particular terminal may lack (e.g., 33 teletypes have no lower case), case shift characters may be defined for upper/lower character/lock shifts and characters may defined which will translate into attention or break (NVT 201), and the synch sequence. The "case.shift.prefix.for", "attention.character=", and "synch.character=" commands are available to independently set each of these characters. In addition, a character may be defined ("quote.prefix" command) to be a single-character quote. The character following this character is always sent regardless of any special action it may otherwise have.

If possible, case shift characters will be used to indicate the case of both input and output. Thus the case shift characters may not be echoed when typed but rather before the output.

All special characters are listed by the "current.modes.are" command. This includes the escape character and the clear output buffer character.

8.10 LEAVING TELNET

To leave Telnet, it is first necessary to return to command mode by typing the escape character. This is because while in remote mode all characters except the escape character are passed on to the remote host or modify characters passed to the remote host. Once in command mode, you may return to the EXEC by typing control-C (ETX) or by using the "quit" command. Continuing from the EXEC will resume with no loss. The "logout" command will disconnect from any remote job and logout your local job. The "exec" command will start up an inferior EXEC under Telnet. From this inferior EXEC, it is possible to perform assemblies or any other task involving the running of subsystems. The "run" command allows an arbitrary program to be run in an inferior fork of Telnet. The "run" may be interrupted by the Telnet escape character.

8.11 MULTIPLE CONNECTIONS

Telnet provides a facility for multiplexing a user's terminal among several remote jobs, thus allowing several simultaneous activities. This is done by giving a name for each connection as it is created. The user may specify the name, or Telnet will default the name to a number. The "retrieve.connection..." command causes the named connection to be made current and remote mode to be entered. Noncurrent connections remain active, but any output received is buffered until that connection again becomes active. Terminal input goes only to the currently active connection.

Telnet may be made to announce the receipt of output on a noncurrent connection with the "signal.waiting.output" command; it may also be caused to hunt for and switch to any active connection -- for more information see "wait.for.any.active.connection" and "auto.switch.to.active.connection" commands.

The name of the current connection may be changed after it is established by means of the "name.for.current.connection" command. The name so specified may be up to 6 characters in length and must be unique.

8.12 TYPESCRIPT FILE

Telnet provides a means of saving on a file a copy of the typescript for a session. This is useful for producing hard copy of the session when using a scope terminal or for producing documentation of procedures or demonstrations. Telnet is started with no typescript file assigned. The "typescript.to.file" command may be used to assign one. The typescript consists of a nearly exact copy of what appears on the terminal with the exception of that which occurs during the execution of the "exec" or "run" or "ddt" commands. "Nearly" refers to slight differences in the spelling of file names in certain Telnet commands. For privacy, the typescript file is given a protection that allows no access to anyone but "self".

8.13 DIVERTING OUTPUT

The output stream may be diverted to some other file with the "divert.output.to.file" command. While diverting output, Telnet sends all output to the indicated file and sends a line to the terminal only when the terminal's output buffer is empty. Thus the terminal monitors the transmission of the stream to the file. The diverted output consists only of characters from the remote host. Telnet commands and responses do not appear in the diverted information. This mode is useful as a primitive file transfer mechanism or to

allow printing of large amounts of terminal output to be done with the lineprinter. It is cancelled by "no divert.output ...".

8.14 INPUT FROM A FILE

The input stream to a remote job may be taken from a file instead of the local terminal by means of the command "take.input.stream.from.file". Telnet blocks terminal input to the connection current when the file is specified, and transmits characters from the named file (echoing as usual according to current modes). However, input to other connections and in command mode is from the user's terminal. When the given file reaches EOF the file is closed and released, and input reverts to the terminal. The user may also manually cancel file input by escaping to command mode and giving "no take.input ...". This mode is useful for routine sequences performed in the remote job. Note that a connection must be established and current when input to it is diverted to a file. Note also that file input is suspended when TELNET is returned to local mode or when another connection is made active; it is not possible to let an input file run while attending another connection.

8.15 TELNET COMMAND SUMMARY

#Connection.to <host> or host name

Performs ICP to connect to the indicated host. Options are available for specifying initial connection socket name or number, and initializing modes from the mode file via the following subcommands. Note that if <host name> is used as a command, only the NAME of a SERVER host may be given (e.g., BBN-TENEX). The argument for "Connection.to" may be any host name or an octal host number.

<octal number>

An ICP is performed to connect to the indicated service socket. Normally socket 1 is assumed.

Logger

Sets socket to 1.

Wait

The connection attempt is repeated until successful.

Name.for.connection.is <name>

Sets the name for this connection as specified.

[no] load.settings

Determines whether to use current mode settings or to load new ones from the mode file.

#Disconnect <cr>

Disconnects the current connection. This will not necessarily log you out from the remote host. Perform the necessary operations before disconnecting.

#Disconnect <name>

Disconnects the connection with the specified name.

#Net.exec

Connects to BBN socket 15600031 wherein the RSEXEC (Resource-Sharing Executive) is found.

#Status.of <host>

Performs ICP with the indicated host and prints its status.

#Echo.mode.is

Sets echo mode according to the following subcommand.

[no] remote

Turns off echoes generated by Telnet and signals the remote computer to

generate echoes. Some hosts are not yet equipped to handle this signal and may require additional action to cause the remote computer to generate echoes. If Telnet believes it is connected to a local half-duplex terminal, it will complain about remote echoes but do it anyway.

[no] local

Turns on Telnet generated echoes and signal the remote computer to not generate echoes. Note that Telnet never generates echoes for terminals it believes have local echo of their own.

[no] linefeed.for.carriage.return

TENEX translates carriage return to EOL, Telnet sends the EOL as the TELNET EOL (i.e., carriage return-linefeed). For some systems, the TELNET EOL is translated into carriage return. For these systems, the appropriate echo is carriage return. Other systems translate the TELNET EOL into carriage return-linefeed. For these systems the appropriate echo is carriage return-linefeed. This subcommand causes the latter echo to be generated.

[no] control.character.echo.for <list of characters>

Turns on local echoes for the indicated control characters. Normally only control-G, J, and M (bell, linefeed, and carriage return) are enabled.

#Terminal.type.is

Allows the user to change Telnet's opinion of his terminal according to the following subcommands. Each command may be preceded by the word "no" to negate its meaning.

Half-duplex

Terminal generates its own echoes.

Full-duplex

Terminal does not generate its own echoes.

[no] lower.case

The terminal has lower case characters.

#Local.mode

If connected, this command prevents Telnet from returning to remote mode after each command.

#Remote.mode
If connected, this command causes Telnet to return to remote mode after each command. If not connected, it does nothing.

#No

May appear before some commands to reverse their action.

#Current.modes.are

Prints the state of connection terminal mode flags, and all special characters.

#[no] character.mode

Causes each character typed to be transmitted as it is typed.

#[no] line.buffer

Causes Telnet to accumulate a line of text before transmitting. A line ends on linefeed or EOL or altmode (esc). The line may be edited with control-A, X, and R.

#[no] raise

Causes lower case letters to be transmitted as their upper case equivalents.

#[no] lower

Causes upper case letters to be transmitted as their lower case equivalents.

#[no] transparent.mode

Causes all characters to pass through Telnet and TENEX untouched. This is needed for special terminals such as the IMLAC using special character stream protocols.

#[no] case.shift.prefix.for

Allows the specification of the four case shift characters according the following four subcommands.

Lock.lower.case

Same as the "Lower" command. Subsequent upper case input will be converted to lower case.

Char.lower.case

Converts the following letter to lower case.

Lock.upper.case

Same as "Raise" command. Subsequent lower case input will be converted to upper case.

Char.upper.case

Converts the following character to upper case.

#[no] unshift.prefix

Causes all following characters to be unshifted, i.e., undoes both an upper case lock and a lower case lock.

#[no] quote.prefix

Causes the following character to be transmitted without regard to any special significance it may have.

#[no] synch.character

The specified character will be converted to the TELNET synch sequence. The TELNET synch sequence is used to cause the remote host examine its input stream to the current point for any special characters (interrupts, attentions, etc.). All nonspecial characters may be thrown away.

#[no] attention.character

The specified character will be converted to the TELNET break or attention character. This character is equivalent to the attention, quit, or break key on certain terminals and may be necessary for using some systems. The !Break! command generates the same character.

#Concise

Turns off automatic command completion. Saves typeout at the expense of readability.

#Verbose

The opposite of concise.

#[no] fancy.command.interpret

Commands are checked character by character. If a character does not fit, it is ignored and not echoed (full duplex terminals only).

#[no] divert.output.stream.to.file

Causes all subsequent output from the remote computer to be written on the specified file. Use "No divert..." to stop this.

#[no] take.input.stream.from.file

Causes subsequent input to the remote host on the current connection to be read from the specified file; input to other connections and in command mode is still from the user's terminal. File is automatically closed and released at EOF; user may force this by "No take.input...", after escaping to command mode. File input, like terminal input to a connection, is active only in remote mode and when connection is current.

#[no] typescript.to.file

A record of the session is kept on a file including both input and output. This is useful for providing hard copy with scope terminals.

typescript.to.file <cr>

The file kept is TELNET.TYPESCRIPT;S in the LOGIN (not connected) directory.

typescript.to.file <filename> <cr>

The named file receives the typescript.

no typescript.to.file

The typescript file (if any) is closed and released; subsequent terminal activity is not saved.

#Escape.character=

The specified character becomes the Telnet escape character. This character must be a TENEX interrupt character. "?" will type what these are.

WARNING: If you have anywhere in your programming a control-Z you should change your escape character in TELNET to other than control-Z to avoid mishaps.

#Clear.output.character=

The specified character becomes the clear output buffer character. Typing this character generates an interrupt which causes the terminal output buffer and any accumulated output to be cleared.

#Help

Prints the file <DOCUMENTATION>TELNET.HELP on the user's terminal.

#Describe <identifier>

Looks up the given identifier in the Help file and prints the accompanying description: an efficient way to read the Help text. Type "describe ?" to get a list of identifiers; command recognition operates on input of identifier.

#Netstatus

Runs <SUBSYS>NETSTAT.SAV.

#Socket.map

Prints a list of all current connection on the system. Optional arguments may be used to select a particular host and a particular connection state.

#Run

Runs the specified file. Like the EXEC's run command.

#Quit

Returns from Telnet to the superior fork (usually the EXEC). May be continued with no loss.

#Logout

Logs out the local job (not the remote one). Requires confirmation with a carriage return.

#Reset

Re-initializes Telnet producing an essentially virgin copy.

#Ddt

Enters ddt. If ddt is not loaded, this will result in an unexpected interrupt. No harm is done if this happens.

#Exec

Starts up an inferior EXEC under Telnet. This EXEC may be used like an ordinary EXEC to run subsystems etc. without disturbing any existing connections. The Telnet escape character will return to Telnet, however.

#Code

Transmits the character specified by the argument. The argument is a taken as an octal number unless preceded by "d" for decimal or "h" for hexadecimal. The argument may be preceded by "o" for octal.

The "code" command argument may be used as a command by itself and will cause the indicated code to be transmitted.

#!break!

Transmits the TELNET break character.

#!synch!

Transmits the TELNET synch sequence. Occasionally the "!synch!" command will work where the synch character will not, since the command bypasses the buffering which may interfere with the use of the synch character.

#Write.modes.for.host

Causes the current mode flags to be saved on the <SYSTEM>TELNET.MODES file under the specified host. Requires write access to the file and is thus not available to ordinary users.

#Retrieve.connection.under.name

Retrieves the connection previously saved under the specified name.

#Wait.for.any.active.connection

Used with multiple connections to wait for and switch attention to the next connection that has any output waiting. Useful when several independent tasks are being run and you wish to know when one completes and switch to that task.

#[no] auto.switch.to.active.connection

Used to switch between tasks on several connections which may each be inactive for long periods. If the current connection is inactive on both input and output for a given number of minutes, Telnet will begin to hunt for any other active connection. If and only if one is found, that connection is made current. The "inactivity time constant" may be specified as any positive integral number of minutes if the "auto.switch..." command is terminated by a space. A <cr> terminator invokes the default value of 2 minutes. "No auto.switch..." disables this feature (current connection remains current until manually changed).

#Where.am.I

Prints a summary of the local job, system, user, terminal and the remote host and socket.

#[no] Signal.waiting.output

Causes all noncurrent connections to print a message when output becomes available.

#Host.names

Lists all current host names with corresponding octal host numbers.

#List.connections

Lists the name, local socket, foreign host, and foreign socket of all connections.

#Flush.host

Marks all connections to the specified host as dead and sends a reset to that host. Requires wheel or operator special capability.

#Comments

An initial semicolon causes the remainder of the line to be ignored. Useful for comments or typing to links.

9. RESOURCE SHARING EXEC (RSEXEC)

The Resource Sharing Executive (RSEXEC) is an evolutionary multicomputer executive program. It provides an environment in which the range of many features found on a single-host time sharing system are extended beyond the boundaries of a single host to encompass many hosts on the ARPANET. At present RSEXEC includes facilities for inter-host user-user interaction (see descriptions for WHO, WHERE, SITES, LINK, SNDMSG), for managing "multi-host" file directories (see descriptions of ENTER and BIND) and for controlling multiple "jobs" on several hosts (see descriptions for TRANSACTION and INITIATE). In addition, the RSEXEC serves as a command language interpreter for TIP users. The DESCRIBE command can be used to obtain descriptions of all (accessible) RSEXEC commands and, in addition, the following terms:

```
BOUND-DEVICE
COMPONENT-DIRECTORY
COMPOSITE-DIRECTORY
FILE-NAMES
INTERRUPT-CHARACTERS
MULTI-IMAGE-FILES
PRIMARY-DIRECTORY
PROFILE
DEFAULT-COMPONENT-DIRECTORY
TRANSACTION
BUGCHK
```

(TIP users accessing RSEXEC via the TIP "@n" command can use only a subset of the RSEXEC commands; they can obtain descriptions of only those commands (and related terms) they have access to.) The user interested in the design philosophy of RSEXEC and its implementation is referred to the paper "A Resource Sharing Executive for the ARPANET", *Proceedings of 1973 National Computer Conference and Exposition*, (also NIC 14689).

```
"?" gives a list of commands.

Use the "DESCRIBE" command to obtain descriptions of other commands.

Only enough of a command to uniquely identify it need be typed.

"ESC" invokes command recognition and completion.
```

Editing characters are:

↑A (Control A) - Character delete.

↑R (Control R) - Retypes current line or item.

↑W (Control W) - Word delete.

RUBOUT (or DEL) - Aborts current command (if typed while still giving *command or arguments).*

↑C (Control C) and ↑T (Control T) are handled by RSEXEC.

↑P (Control P) may be used as a panic escape in case your terminal becomes hung while linked. It breaks the link, clears input and output buffers, and returns to the higher level EXEC. The CONTINUE command will then resume the RSEXEC session as if a ↑C had occurred.

9.1 COMMAND SUMMARY

```
ACQUIRE (Component Directory) Component1 ... Componentn <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Used to add the files in the component directories specified to the composite directory. * may be used in the component directories. *A directory need not be ACQUIREd in order to* reference files in it or at the corresponding site. However, file name recognition and completion will work only for files that are local or in ACQUIREd directories. See also descriptions for COMPOSITE- DIRECTORY, COMPONENT-DIRECTORY, and RELEASE.

```
APPEND (file) FILE1 (to file) FILE2 <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Changes FILE2 by appending FILE1 to it. The name and extension of FILE2 default to same as FILE1 if recognition (ESC,↑F) is used.

```
BIND (device) DEVICE-NAME (to site) SITE-NAME <cr>
or
BIND (device) DEVICE-NAME (to site) TIP-NAME <cr>
  -(Port  ) number <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Associates the device name with the host name such that subsequent references to the device name refer to the device at the host specified. For example, the sequence

```
-BIND LPT MITRE-TIP
--(Port  ) 1
-LIST TEST.DATA
```

would produce a listing of file TEST.DATA at device port 1 on the MITRE-TIP. For binding to a TIP port to work properly the TIP port in question must be set to "wild".

BOUND-DEVICE

The user can use the BIND command to specify that subsequent use of a particular device name is to refer to that device at a specific site. Such a device is said to be "bound" to that site. For example, the sequence of commands

```
-BIND LPT USC-ISI <cr>
-COPY REPORT.DRAFT LPT: <cr>
-LIST PROGRAM.SOURCE <cr>
```

first binds the line printer to ISI and then causes two listings to be produced by the ISI line printer.

BREAK<cr>

Breaks terminal links (see LINK).

BUGCHK

RSEXEC contains a considerable number of internal consistency and redundancy checks. If RSEXEC detects a malfunction it prints the message

```
-BUGCHK at NNNNN
```

and continues. Such messages are useful for debugging purposes, recurring BUGCHK messages, together with the circumstances under which they occur, should be reported to Bob Thomas (BTHOMAS@BBN) or Paul Johnson (JOHNSON@BBN).

COMPONENT-DIRECTORY

One of the file directories that may be included in the user's composite directory. The syntax for component directories is:

```
[Site]<Directory>
```

e.g., [NIC]<JONES>. There is an entry in the user's profile for each component directory. The user may control which component directories are, at any given time, included in his composite directory via the ACQUIRE and RELEASE commands and the subcommands of the ENTER command. * (to indicate all such items in the profile) may be typed for the site or directory in any command that accepts more than one component directory.

COMPOSITE-DIRECTORY

The collection of file directories specified in a user's profile define his composite directory. The "contents" of the composite directory are the union of the "contents" of the component directories specified in the profile. Pathnames without site and directory qualification are interpreted with respect to the user's composite directory. The ENTER command uses information in the profile to gather sufficient information to construct (a local copy) the user's composite directory. See also descriptions for PROFILE and FILE-NAMES.

CONTINUE <cr>

Resumes execution interrupted by previous ↑C.

COPY (file) FILE1 (to new file) FILE2 <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Makes a copy of FILE1 which is named FILE2. The name and extension of FILE2 default to same as FILE1 if recognition (ESC,↑F) is used.

DDT <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command. Causes DDT to be loaded (if necessary) into user fork address space and starts DDT debugger.

DEFAULT COMPONENT DIRECTORY

When a new file (or a new version) is to be created and no component directory has been specified, the file is created in the default component directory. The default component directory is initialized when the ENTER command is given to be the user's local login directory. It may be changed by the DEFAULT command of the profile editor (PROEDIT) or the DEFAULT subcommand of the ENTER command. The default is not(!) stored in the permanent profile.

DELETE (file) FILE <cr>
or
DELETE (file) FILE1 ... FILEn <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Deletes the file(s) specified. * may be used. If recognition (ESC,↑F) is used, the name and extension of each file after the first default to those of the previous file. Files which have been deleted but not expunged may be "undeleted" by the UNDELETE command. Deleted files are automatically expunged at LOGOUT.

```
DESCRIBE (command, term or ALL) command<cr>
or
DESCRIBE (command, term or ALL) ALL<cr>
```

Describes any (or all) command(s). In addition, DESCRIBE can be used to describe certain "terms" such as RSEXEC.

```
DEVSTAT <cr>
```

Lists the (binding) status of all devices.

```
DIRECTORY <cr>
or
DIRECTORY , <cr>
 -subcommand <cr>
    .
    .
    .
 -<cr>
or
DIRECTORY FILE1 ... FILEn <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Prints information (as modified by subcommands, if any) about the file(s) specified or, if none are specified, all files in the user's composite directory. * may be used in the file names. Version defaults to * if omitted; name and extension default to * only if $(altmode,ESC) or ↑F(Control-F) is used for recognition.

The subcommands are:

```
--VERBOSE <cr>
  prints the file's location, size, write date, and read date
```

```
--MULTI-IMAGE FILES <cr>
  only reports on multi-image files
--SITES site1 ... siten <cr>
  only reports on files at the specified site(s)
--DELETED-FILES <cr>
  reports deleted files instead of undeleted ones
--IMAGES
  prints location(s) of the file's image(s)
```

See also descriptions of COMPOSITE-DIRECTORY and MULTI-IMAGE-FILES.

```
ENTER (name) NAME (affiliation) AFFL (RSEXEC password) PWRD <cr>
or
ENTER (name) NAME (affiliation) AFFL (RSEXEC password) PWRD , <cr>
-subcommand <cr>
   .
   .
   .
-<cr>
```

Grants access to distributed file system features of RSEXEC after constructing a composite directory for the user from his profile. If the user does not have a permanent profile (e.g., hasn't previously used the ENTER command or has chosen not to have RSEXEC maintain a permanent profile for him) RSEXEC will acquire the information necessary to construct a profile for him. NAME is the name the user has chosen to be known by to RSEXEC; AFFL is his affiliation (e.g., AMES, NIC, ARPA - at present an arbitrary string); PWRD is his RSEXEC password chosen when his permanent profile was created. The user may use the ENTER subcommands to control which components of his profile are acquired for his composite directory.

The subcommands are:

```
ACQUIRE COMPONENT1 ... COMPONENTn
     the specified component directories will be acquired

RELEASE COMPONENT1 ... COMPONENTn
     the specified component directories will not be acquired

DEFAULT COMPONENT
     sets Default Component Directory to COMPONENT
```

may be used in the ACQ and REL commands. See also the descriptions for PROFILE, COMPONENT-DIRECTORY, COMPOSITE-DIRECTORY, ACQUIRE, DEFAULT-COMPONENT-DIRECTORY, and LPWCHANGE.

ESCAPE (Character is) CNTL-CHAR <cr>

Sets the "return from transaction escape character" to the control character specified. The escape character is initially ↑Z. See also the descriptions for INTERRUPT-CHARACTERS and USE.

EXEC<cr>

Runs the standard TENEX EXEC; to return to RSEXEC use the EXEC QUIT command. If he has previously ENTERed, the user has the option of reacquiring the local component(s) of his composite directory when he returns to RSEXEC from an inferior EXEC. This is useful if he has added or deleted files while using the EXEC.

EXPUNGE (deleted files) <cr>

Irretrievably removes deleted files from the user's composite directory and from each component directory for which such a deleted file is found in the composite directory.

FILE-NAMES

The RSEXEC extends the syntax for TENEX file names to include a Host component. The syntax for file pathnames is:

[HOST]DEVICE:<DIRECTORY>NAME.EXTENSION;VERSION

Where HOST is either the string "LOCAL" or the name of an ARPANET TENEX. Partial pathnames may be used within RSEXEC. For example, whenever the site, device and directory fields are omitted, the user's composite directory is used as a default. At present the TENEX "*" convention may be used only in the name, extension, or version for local files or files in the composite directory. The user must have a profile entry for a site before he can access files at that site. A Component Directory for a site need not be ACQUIREd in order to reference files at the site or in the Component Directory. However, file name recognition and completion will work only for files that are local or in an ACQUIREd directory. See description for PROFILE.

FILE-TRANSFER-EXAMPLES

The NEED command is a convenient way to move files from one or more Hosts to a specified destination Host: (assume in the following that the local Host is BBNA):

To move a group of files to the default component directory:

-NEED (files) F1 F2 ... FN <cr>

e.g.,

-NEED (files) [ISI]<SUBSYS>NETSTAT.SAV [BBNB]<TENEX-132>SCHED.MAC<cr>
[ISI]<SUBSYS>NETSTAT.SAV...OK
[BBNB]<TENEX-132>SCHED.MAC...OK

To move a group of files to a directory other than the default component directory:

-NEED (files) F1 F2 ... FN, <cr>
--(in Component Directory) CD <cr>

e.g.,

-NEED (files) [ISI]<SUBSYS>NETSTAT.SAV [BBNB]<TENEX-132>SCHED.MAC ,
--(in Component Directory) [ARC]<JONES>
[ISI]<SUBSYS>NETSTAT.SAV...OK
[BBNB]<TENEX-132>SCHED.MAC...OK

Note that component directories need not be "ACQUIREd" to move to or from them. However, file name recognition and completion will work only for files that are local or in ACQUIREd directories.

FILSTAT <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command. Similar to the EXEC FILSTAT command. Prints status of files currently in use by the user.

FULLDUPLEX<cr>

Causes your terminal to be treated as fullduplex.

GET (Saved File) FILE <cr>

*****Not Implemented*****

HALFDUPLEX<cr>

Causes your terminal to be treated as halfduplex.

HELP<cr>

Prints a short help message.

HOSTAT<cr>

Lists the status of network server hosts as maintained by the host survey program at MIT-DMCG.

INITIATE (transaction at) HOST-NAME (called) NAME <cr>

Attempts to create a job for the user at the site specified. The job is known as NAME. The user will be notified when the transaction is ready for use. See also the descriptions for the USE, TERMINATE, TRSTAT and PURGE commands.

INTERRUPT-CHARACTERS

The following characters are handled as terminal interrupts by RSEXEC:

> ↑C (CNTL-C): interrupts the current activity, returning control to RSEXEC. The CONTINUE command may be used to resume the interrupted activity. when a transaction is being USEd, RSEXEC transmits the ↑C to the remote transaction.
>
> ↑T (CNTL-T): prints CPU and console time used in RSEXEC session. When a transaction is being USEd, RSEXEC transmits the ↑T to the remote transaction.
>
> ↑Z (CNTL-Z): enabled only when a transaction is being USEd. Returns control from transaction to RSEXEC. The ESCAPE command may be used to change the transaction escape character from ↑Z to another control character.
>
> ↑P (CNTL-P): RSEXEC "panic" escape. Intended for use when your terminal becomes "hung." It breaks all terminal links, clears terminal input and output buffers, and returns control to the top level EXEC. The EXEC CONTINUE command may be used to resume the RSEXEC session. When resumed in this way the RSEXEC acts as if the user had typed ↑C.

LEAVE (distributed file system) <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Makes the distributed file system features of the RSEXEC inaccessible. Inverse of ENTER.

LINK (to) user name or terminal number (at Host) hostname<cr>
or
LINK (to) user name or terminal number<cr>

"Links" your terminal to the user or terminal specified at the host specified such that the output for either terminal appears on both. If no hostname is given, the local host is assumed and a local link will be made. The RSEXEC comment character (;) should be used when LINKed to prevent RSEXEC from interpreting dialogue as commands.

Links are broken by the BREAK command or by quitting RSEXEC.

LIST (file) FILE <cr>
or
LIST (file) FILE1 ... FILEn <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Causes a listing(s) of the specified file(s) to be output to the line printer. * may be used. If recognition (ESC,^F) is used, the name and extension of each file after the first default to those of the previous file. The command

```
-COPY (file) FILE (to new file) LPT: <cr>
```

will produce a listing without the formatting action of the LIST command.

LOCATE

The "LOCATE" command has been discontinued. See instead the description of the "NEED" command.

LOGOUT<cr>

Logs out from RSEXEC and TENEX.

LPWCHANGE (old local password) OLDP (current local password) NEWP <cr>

Notifies RSEXEC that your local TENEX password has changed. This makes it possible to use a profile (via ENTER) that was created before your TENEX password changed. If the command isn't given, such an ENTER will fail. Once the ENTER succeeds, the ENTERed profile will automatically be modified to reflect the password change.

MEMSTAT <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

MEMSTAT is similar to the EXEC MEMSTAT command. It prints a summary of the address space of the process currently assigned to the user (for execution of subsystems or PRIVate programs.

MULTI-IMAGE-FILES

The RSEXEC treats files with the same pathname relative to a user's composite directory (i.e., identical name, extension and version components) as "images" of the same file. Such

a file is said to be a multi-image file. Although the profile file (see description of USER PROFILE) is transparent to the RSEXEC user, it is implemented as a multi-image file.

```
NEED (file) FILE1 ... FILEn <cr>
or
NEED (file) FILE1 ... FILEn , <cr>
 -(in Component Directory) CD1 ... CDn <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Creates an image of each file specified in each component directory specified. If no Component Directory is specified, the image(s) is placed in the default component directory. * may be used in the component directories and in the file names. If recognition (ESC,^F) is used, the name and extension of each file after the first default to those of the previous file. See also descriptions of MULTI-IMAGE-FILE, COMPONENT-DIRECTORY, and DEFAULT-COMPONENT-DIRECTORY.

```
NETNEWS<cr>
```

Prints the latest network news.

```
NETSTAT<cr>
```

Runs the standard TENEX NETSTAT subsystem which gives network status information.

```
PRIMARY-DIRECTORY
```

For each site for which there are Component Directories there is a Component Directory designated the Primary Component Directory for that site. The Primary Directory for a site must be a "login" (rather than "files only") directory. It is used as the basis for access control checks at the site. The Primary Directory for a site is set by the user either implicitly (as the first "login" directory for the site added to his profile) or explicitly (via the PRIMARY command of the profile editor). See also descriptions for PROFILE, PROEDIT and COMPONENT-DIRECTORY.

```
PROEDIT <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Used to edit the user profile. PROEDIT subcommands are:

```
ADD       to add an entry to the profile
REMOVE    to remove an entry from the profile
LIST      to print the profile
CHANGE    to modify an existing profile entry
DEFAULT   to change the default component directory
PASSWORD  to change the RSEXEC password used to ENTER the profile
PRIMARY   to change the primary directory for a site
UPDATE    to make the edits permanent (see below also)
QUIT      to return to the RSEXEC
```

The syntax for a profile entry is:

```
[Site]<Directory>
```

If successful, the ADD (REMOVE) command results in modification of the user's profile with the addition (removal) of the specified entry. In addition, if he has so chosen, his composite directory is modified by the addition (removal) of the appropriate files. If the user attempts to ADD an entry for a site for which the RSEXEC server program is down, the entry will be marked to indicate that it has not been verified (i.e., name/password not checked) and that its files have not been added to the user's composite directory. If during the course of the user's RSEXEC session the remote server comes up, the entry will automatically be verified and its files added to the user's composite directory (if the user has indicated that he wants them). The user has the option of making his edits permanent either via the UPDATE command or when he leaves the distributed file system environment via the LEAVE, QUIT or LOGOUT command. The profile editor prompt character is "*". See also the descriptions for PROFILE, COMPONENT-DIRECTORY, DEFAULT-COMPONENT-DIRECTORY, COMPOSITE-DIRECTORY, PRIMARY-DIRECTORY.

PROFILE

A collection of user-specific information and parameters maintained for the user by the RSEXEC. At present, the information maintained includes an entry for each of the user's file directories: each entry consisting of Host name, directory name, password and account number or string. The profile editor (PROEDIT) can be used to add or delete entries from the profile. If a user chooses to have the RSEXEC maintain a permanent record of his profile a file named

```
]-RSPRF-[.NAME@AFFILIATION;1
```

will be maintained in each directory named in the profile. This file is itself transparent to the RSEXEC user. Images of the profile file are suitably protected; only the user himself may read or write it (its protection attribute is P770000); the passwords stored in it are encrypted (using the user's RSEXEC password as a key). The QUIT, LEAVE and LOGOUT commands ask the user if he wishes to have a permanent profile.

PROLIST (Component Directory) COMPONENT1 ... COMPONENTn <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Prints the specified profile entries, or the entire profile if none are specified. * may be used in the component directory specifications. See also description of COMPONENT-DIRECTORY.

PURGE (transaction) NAME <cr>

Causes forced termination of a previously INITIATEd job or TELCONN connection by breaking network connection with remote site. Intended for use only when TERMINATE fails. See also descriptions for INITIATE, TELCONN, USE, TERMINATE, and TRSTAT.

QFD <cr>
or
QFD FILE1 ... FILEn <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Prints a "quick" description of the file(s) specified. Same as DIRECTORY but prints the file's location, size, and dates and times of create, write, and read. See description of DIRECTORY.

QUIT<cr>

Ends RSEXEC session.

RECEIVE (links)<cr>

Sets terminal to accept links (default state). Undoes a previous REFUSE command.

REFUSE (links)<cr>

Sets terminal to refuse links. Undone by a subsequent RECEIVE command.

RELEASE (Component Directory) Component1 ... Componentn <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command. Used to remove from the composite directory the files in the specified component directories. * may be used in the component directories. See also the descriptions for COMPOSITE-DIRECTORY, COMPONENT-DIRECTORY, and ACQUIRE.

RENAME (file) FILE1 (to be) FILE2 <cr>

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Changes the name of FILE1 to be FILE2. The name and extension of FILE2 default to same as FILE1 if recognition (ESC,↑F) is used.

RESET <cr>

Similar to the RESET command of the TENEX EXEC.

RUN (Saved File) FILE <cr>

*****Not Implemented*****

SAVE (Core From) FFF (To) TTT (On) FILE <cr>

*****Not Implemented*****

SERVERS<cr>

Prints a list of the sites which (at times) run RSEXEC servers. These sites must both be up and running the server to be accessible from RSEXEC.

SITES (of user) username<cr>

Lists the sites (with RSEXEC servers running) at which the specified user is known.

SNDMSG<cr>

Runs a subsystem for sending messages to other network users. Messages can be delivered only if the destination site runs an FTP server with the MAIL command implemented. Undelivered messages will be deleted after a week.

SSAVE (Pages From) FFF (To) TTT (On) FILE <cr>

*****Not Implemented*****

START <cr>

*****Not Implemented*****

```
TELCONN (to site) HOST-NAME (with a connection called) NAME<cr>
or
TELCONN (to site) HOST-NAME (with a connection called) NAME,<cr>
 -option <cr>
 -  .
 -  .
 -  .
 -<cr>
```

Attempts to establish a TELNET connection to the specified HOST. If successful, the user's terminal is connected to this network communication path. To return to RSEXEC type ↑Z (CNTL Z). Use of the connection may be resumed with the USE command. Default socket is the logger (#1). An alternate socket, as well as desired echo mode and terminal characteristics, may be specified in the second form of the command. Type ? in response to the -- option prompt to list available options. See also the descriptions of the USE and INITIATE commands.

```
TENXSTAT<cr>
```

Prints status information for TENEX sites with RSEXEC servers running.

```
TERMINATE (transaction) NAME <cr>
```

Terminates a previously INITIATEd job by sending it several ↑C's and then logging it out. Also can be used to terminate a TELCONN connection. After termination, a TELCONN connection can no longer be USEd.

```
TIMECONSTANT (for net connections is) value<cr>
```

Sets the time constant used for interactions with non-local RSEXEC server programs. If the remote server does not respond within the specified time, the interaction is aborted. Possible values are: RAPID (8 sec.), MODERATE (15 sec.), LETHARGIC (40 sec.), and INFINITE (2 min.). The time constant is initially MODERATE (15 sec.).

```
TRANSACTION
```

A user can instruct the RSEXEC to create a job for him at another site. Such jobs are called transactions. See descriptions of the INITIATE, TELCONN, USE, TERMINATE and PURGE commands.

```
TRSTAT <cr>
```

Prints status of previously INITIATEd jobs and TELCONN connections. Possible status conditions are:

```
PENDING              INITIATEd but login incomplete
USABLE               can be used via USE command
USABLE TELNET        TELCONN connection, can be used via USE
                     command
TERMINATION PENDING  TERMINATEd but logout incomplete
TERMINATED           TERMINATEd but not yet removed from
                     RSEXEC's transaction table
```

```
TYPE (file) FILE <cr>
or
TYPE (file) FILE1 ... FILEn <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Prints the file(s) specified on the user's terminal. Identical to LIST command except for the use of the terminal (see description of LIST). The command

```
-COPY (file) FILE (to new file) TTY: <cr>
```

will copy a file to the user's terminal without the formatting action of the TYPE command.

```
UNDELETE (file) FILE <cr>
or
UNDELETE (file) FILE1 ... FILEn <cr>
```

Use of this command is limited to users who have gained access to the file system features of RSEXEC via the ENTER command.

Revives the DELETEd file(s) specified by restoring the file(s) to normal status; e.g., they are once again accessible to RSEXEC commands. The inverse of DELETE. * may be used. If recognition (ESC, ↑F) is used, the name and extension of each file after the first default to those of the previous file.

```
USE (transaction) NAME <cr>
or
USE (transaction) NAME, <cr>
 -option <cr>
 -  .
 -  .
 -  .
 -<cr>
```

Connects the user's terminal to a previously INITIATEd job or to a previously established TELCONN connection. To return to RSEXEC type ↑Z (CNTL-Z); to transmit ↑Z to the job type the two-character sequence <↑><Z>. In general typing the two-character sequence <↑><x> will transmit x if x is a nonalphabetic character and CNTL-x if s is an alphabetic. Thus to transmit ↑P (the RSEXEC "panic" escape) type <↑><P>. If the user has ENTERed and uses ↑Z to return to RSEXEC from a transaction, he has the option of updating his composite directory to reflect any additions or deletions resulting from his USE of the transaction. The ESCAPE command may be used to change the transaction escape character from ↑Z. If the second form of the command is used, options concerning the network connection (echo modes, terminal characteristics, etc.) may be specified. To list the allowable options type "?" when prompted with "--". While USING any TELNET connection (created by either INITIATE or TELCONN) certain TELNET control functions may be invoked by typing the TELNET dynamic option escape character followed by a character indicating the desired function. The dynamic option escape character is initially ↑D (CNTL D) but may be changed using the DYNAMIC option. The recognized commands are:

```
<↑D><L> means to do Local echoing
<↑D><R> means to do Remote echoing
<↑D><T> means to Transmit accumulated chars immediately
<↑D><B> means send TELNET Break character
<↑D><S> means send the TELNET Synch sequence
Any other character will be transmitted as data.
To send a  <↑D> character through the network, type <↑><D>.
```

```
WHERE (is user) username<cr>
```

Lists all active jobs belonging to the specified user at all sites (with RSEXEC servers running).

```
WHO<cr>
or
WHO (at site) hostname<cr>
```

Lists users with active jobs at specified (or all) network site(s) with RSEXEC servers running.